Departamento de Informática

PROYECTO FIN DE CARRERA

APLICACIÓN WEB PARA LA GESTIÓN, REPARTO Y ASIGNACIÓN DE TAREAS GENÉRICAS

Autor: EDUARDO JORGE GÓMEZ

Tutor:

Leganés, Octubre de 2015

Título: APLICACIÓN WEB PARA LA GESTIÓN, REPARTO Y ASIGNACIÓN DE TAREAS GENÉRICAS

Autor: EDUARDO JORGE GÓMEZ

Director:

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de Octubre de 2015 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

Muchas gracias a Loli, mi mujer, por tu apoyo, por tu insistencia, por tu cariño y por estar ahí en todo momento. Gracias a ti he reunido las fuerzas necesarias para afrontar este reto. Sin tu ayuda no hubiera sido posible. Tus palabras de apoyo han hecho que este camino lo haya recorrido a favor del viento.

María, Bea, Víctor, Inés y Mohsin, muchas gracias por el apoyo y los ánimos que me habéis dado, gracias a ellos sabía en todo momento que no caminaba solo.

Resumen

El desarrollo de una aplicación destinada a la gestión, reparto y asignación de tareas se plantea partiendo de las necesidades reales que se observan en numerosos ámbitos del mundo empresarial. Muchas empresas dedicadas a realizar tareas suministradas por terceros tienen infinidad de problemas a la hora de llevar un control de los distintos estados de dichas tareas y hacer un reparto eficiente entre sus trabajadores.

Esta aplicación web es una aplicación para la gestión de tareas genéricas que incluye un módulo de reparto y asignación entre grupos de trabajo. La aplicación tiene muchas funcionalidades y es ampliamente parametrizable. La aplicación está diseñada para facilitar a las personas y a los equipos la gestión de sus tareas y el reparto de ellas entre el grupo de trabajo basado en criterios de búsqueda y ordenación. La aplicación se basa en el concepto de Modelo de Trabajo, entendiéndolo como una abstracción de cada clase de trabajo a realizar.

Palabras clave: tareas, reparto de tareas, colas de trabajo, asignación

Abstract

The development of an application for the management, allocation and assignment of tasks arises based on the real needs observed in many areas of the business world. Many companies dedicated to perform tasks provided by third parties have many problems when it comes to keeping track of the different states of these tasks and make efficient sharing among its workers.

This web application is a generic management tasks including module division and allocation between workgroups. The application has many features and is widely programmable. The application is designed to help individuals and teams manage their tasks and the distribution of them among the working group based on search criteria and management. The application is based on the concept of Working Model, understood as an abstraction of every kind of work to be done.

Keywords: work, work sharing, job queues, assignment

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y objetivos

1.1 Introducción

Muchas empresas dedicadas a realizar tareas suministradas por terceros tienen infinidad de problemas a la hora de llevar un control de los distintos estados de dichas tareas y hacer un reparto eficiente entre sus trabajadores.

No existen en el mercado demasiadas aplicaciones que solucionen el problema de manera genérica. Debido a esto se plantea la realización de una aplicación que intente dar solución de forma genérica a los problemas planteados.

El objetivo de la aplicación planteada en este proyecto es solucionar las necesidades de control de los estados de las tareas y asignación de las mismas de forma eficiente entre los diferentes grupos de trabajo existentes, todo ello de una manera genérica y sencilla con un alto grado de parametrización[1]. Para ello se contemplarán las posibilidades tanto de asignación manual como de reparto automático basado en criterios predefinidos.

Este proyecto pretende llevar a cabo, como primer paso, una exposición de los problemas con los que se encuentran numerosas empresas dedicadas a realizar tareas suministradas por terceros, referidos tanto al control interno de los estados donde se encuentran dichas tareas como la distribución de las mismas dentro de sus grupos de trabajo.

[1] Entendemos por parametrización a la posibilidad de que la aplicación se comporte de una forma u otra dependiendo de los valores asignados a una serie de variables que puede realizar cualquier usuario autorizado para ello

Tras realizar esta exposición de dichos problemas se procederá a realizar un análisis de las aplicaciones existentes en el mercado que pueden dar alguna solución a los problemas planteados.

El siguiente paso, será la presentación de todos los pasos a seguir para la implementación de la aplicación web, desde su primer análisis hasta su finalización. Se procederá a realizar una explicación exhaustiva de todos los parámetros de configuración necesarios para que la aplicación web contemple todas las posibilidades descritas. Además de presentarán ejemplos de funcionamiento de la aplicación web para mostrar al lector una visión completa de su potencial.

Finalmente, se procederá a mostrar el presupuesto elaborado, las conclusiones extraídas tras la realización del proyecto y se realizará una descripción de las posibles futuras líneas de desarrollo a seguir.

1.2 Objetivos

El objetivo fundamental de este proyecto es el de implementar una solución en formato de aplicación web para el problema planteado: Control, asignación y reparto de tareas genéricas de una forma eficiente en función de unos criterios definidos en la parametrización de la aplicación. En base a ese objetivo principal, se proponen los siguientes objetivos parciales:

- Presentar una interfaz amigable que permita la utilización simultánea de la aplicación desarrollada a usuarios con diferentes grados de conocimiento sobre la materia.

- Permitir el acceso a la aplicación desde cualquier dispositivo con un navegador web disponible.

- Permitir el despliegue de la aplicación en entornos distribuidos utilizando el modelo cliente-servidor.

- Ofrecer un alto nivel de parametrización de la aplicación para alcanzar el objetivo de que las tareas sean *genéricas*, es decir que puedan implementar cualquier situación real que se nos plantee.

- Permitir distintos niveles de acceso implementando un sistema de permisos de usuarios basado en perfiles.

- Permitir a los usuarios con los suficientes permisos configurar la aplicación para que se realice un control de las tareas personalizado. Permitiendo a los *administradores* configurar tantos *modelos de trabajo* como sean necesarios. Cada uno de estos *modelos de trabajo* con la información que se desee.

- Permitir a los usuarios con los suficientes permisos configurar la aplicación para que se realice el reparto de tareas basado en criterios de búsqueda y ordenación configurados desde la aplicación

- Permitir a los usuarios *administradores* de los distintos modelos de trabajo llevar un control de la actividad mediante monitores de control.

- Ofrecer la posibilidad de indicar el estado en el que queda la tarea después de su gestión para llevar un seguimiento pormenorizado de la misma.

- Ofrecer la posibilidad de asignación de tareas tanto masiva como individual a usuarios específicos.

1.3 Fases del desarrollo

La realización de este proyecto se descompone en las siguientes fases:

- Ofrecer la posibilidad de asignación de tareas tanto masiva como individual a usuarios específicos.

- Exposición de las necesidades del mercado y planteamiento de la aplicación web como solución.

- Detalle del proceso de desarrollo de la aplicación web y explicación de su configuración y funcionamiento.

- Realización de plan de pruebas y ejecución del plan de implantación de los sistemas necesarios.

- Elaboración de un presupuesto para el desarrollo de la aplicación.

1.4 Medios empleados

Los medios empleados para la realización de este proyecto son los siguientes:

- Editor de textos:

Nombre	Versión
Microsoft Word	**2007**

Tabla 1. Editor de textos usado en la elaboración del proyecto

- Editor de imágenes

Nombre	Versión
Microsoft Power Point	**2007**

Tabla 2. Editor de imágenes usado en la elaboración del proyecto

- Herramientas de desarrollo empleadas en la elaboración de la aplicación:

Nombre	Versión
Java Development Kit (JDK)	**1.7.0_51**
Spring Tool Suite	**3.2.2**
Spring	**3.2.2**
Eclipse Java EE IDE for Web Developers	**Luna M7 Release**

Tabla 3. Herramientas de desarrollo usadas en la elaboración del proyecto

- Herramientas de desarrollo de la interfaz web empleadas en la elaboración de la aplicación:

Nombre	Versión
XHTML	**1.0**
Java Server Faces (JSF)	**2.2**
Framework Primefaces	**5.2**

Tabla 4. Herramientas de desarrollo de interfaz web usadas en la elaboración del proyecto

- Navegadores web

Nombre	Versión
Google Chrome	**45.0.2454.85 m**

Mozilla Firefox	**40.0.3**
Internet Explorer	**10**

Tabla 5. Navegadores web usadas en la elaboración del proyecto

- Herramienta de gestión y construcción de despliegues de la aplicación:

Nombre	Versión
Apache Ant	**1.9.2**

Tabla 6. Herramientas de gestión y construcción de despliegues de la aplicación

- Servidor de aplicaciones:

Nombre	Versión
Apache Tomcat	**7.0.39**

Tabla 7. Servidor de aplicaciones donde desplegar la aplicación

- Servidor de base de datos:

Nombre	Versión
MySQL	**5.6**

Tabla 8. Servidor de Base de Datos de la aplicación

- Herramientas de generación de diagramas de casos de uso:

Nombre	Versión
LucidChart	**Versión web**

Tabla 9. Herramientas de generación de diagramas de casos de uso

- Herramientas de generación de proyecto (diagrama de Gantt):

Nombre	Versión
Smartsheet	**Versión web**

Tabla 10. Herramientas de generación de diagramas de proyecto

1.5 Estructura de la memoria

La memoria de este proyecto ha sido dividida en 5 capítulos cuyo resumen se presenta a continuación:

- En el capítulo 1 "Introducción y Objetivos" se realizará una introducción al proyecto, presentando las motivaciones, estructura y los objetivos que se pretenden alcanzar.

- En el capítulo 2 "Necesidades y planteamiento de la solución" se realizará un análisis de los motivos de la realización de la APLICACIÓN WEB PARA LA GESTIÓN, REPARTO Y ASIGNACIÓN DE TAREAS GENÉRICAS. Además se desglosará las necesidades que debe cubrir la aplicación para una realización eficiente del trabajo a ejecutar en un entorno de producción empresarial.

- En el capítulo 3 "Desarrollo de la Aplicación Web GTareas" se presentará un desglose pormenorizado de las distintas fases en las que se ha dividido el desarrollo de la aplicación web. Además se analizará las distintas funcionalidades que ofrece la aplicación así como las distintas parametrizaciones que ofrece.

- En el capítulo 4 "Plan de pruebas e implantación" se explicará con detalle el plan de pruebas para verificar el correcto funcionamiento de la aplicación. Se desglosarán todas las pruebas realizadas siguiendo el plan establecido. Además se describirá el plan de implantación donde se explicarán los pasos a seguir en el despliegue y puesta en producción de la aplicación.

- En el capítulo 5 "Presupuesto" se describirá, de forma detallada, el presupuesto elaborado para afrontar el proyecto. La descripción del proyecto se apoyará en un diagrama de Gantt y una hoja de cálculo donde quedarán desglosados todos los costes asociados al proyecto.

- En el capítulo 6 "Conclusiones" se expondrán las conclusiones del proyecto, indicando si se han cumplido los objetivos iniciales propuestos.

Capítulo 2

Necesidades y planteamiento de la solución

2.1 Estudio de las necesidades

La información que es tratada dentro del ámbito de una organización empresarial supone un recurso crítico puesto que constituye uno de los elementos clave en el proceso de toma de decisiones.

Es imprescindible que las empresas de servicios, cuyo cometido es la realización de tareas de distinto índole proporcionadas por terceros, dispongan de sistemas informáticos que controlen de forma detallada el ciclo de vida de cada tarea, así como la información referente a dicha tarea.

El control interno en la gestión define un valor añadido en la empresa. La búsqueda de la eficiencia en los procesos internos ha supuesto un incentivo crítico en la mejora de los mecanismos de control interno. Las consideraciones realizadas por las organizaciones a este respecto, hacen que se alcance un nuevo estadio de evolución en el que la mejora de la eficiencia y el control de sus actividades constituyen los objetivos claves a alcanzar.

Es clara la necesidad que tiene el mundo empresarial de la utilización de herramientas informáticas destinadas a llevar un control de las tareas a realizar.

Dentro de dichas tareas es necesario conocer numerosos aspectos en la realización de las mismas. Además es imprescindible dotar tanto al administrador del servicio como al

usuario de una serie de utilidades que hagan sencillo la realización y el control de todas las tareas. A continuación se describen todos estos elementos y utilidades

- Conocer el estado en el que se encuentra cada tarea.
- Historial de los estados por los que ha pasado cada tarea.
- Usuarios que han intervenido cada la tarea y acciones realizadas.
- Posibilidad de asignación masiva o individual de tareas a usuarios.
- Definición de colas de trabajo donde los usuarios se conectarán y se les servirá el trabajo que deben realizar según el criterio predefinido.

Se podría plantear la realización de una herramienta ad-hoc para la gestión de cada tipo de tarea. Debido a la diversidad de tipos de tareas que una empresa de servicios puede llegar a realizar esta solución parece inviable.

Por esto se plantea la necesidad de una herramienta (en este caso una aplicación web) capaz de gestionar las tareas, encargarse de su reparto y asignación, de una forma totalmente genérica y parametrizable.

La herramienta planteada deberá contemplar la necesidad de agrupar las tareas de un mismo tipo en un nivel superior que denominaremos "modelo". Un modelo agrupará todas las tareas similares: estados, campos de información, configuración, etc… Gracias a este nivel superior lograremos almacenar la información de manera más sencilla y estructurada

Además, la herramienta, deberá permitir la asignación de permisos a los distintos usuarios tanto para administrar los tipos de tareas como de operar con las tareas. Se asignarán permisos de administración de modelos para que solo los usuarios con permisos puedan modificar la configuración y la estructuración de estos modelos.

Todo esto deberá ser acompañado de un módulo de monitorización en tiempo de real del estado del servicio. Este módulo de monitorización informará en todo momento al administrador del servicio de la situación del mismo ayudándole a gestionarlo de manera eficiente y facilitarle la toma de decisiones en momentos críticos.

2.1.1 Ejemplo de necesidad de la aplicación web

Para verlo de manera más clara, a continuación se ofrece un ejemplo real donde la aplicación web presentada en este proyecto encajaría perfectamente.

En la actualidad está a la orden del día que las empresas externalicen a empresas de servicios muchos de los procesos que realizan. Por ejemplo, la mayoría de empresas del sector bancario subcontratan a terceros la realización de una serie de procesos para reducir sus costes de producción.

Las empresas de servicios reciben la "cartera de trabajo" (lista de tareas a realizar dentro del proceso externalizado) mediante diversos canales de comunicación. Estos canales de comunicación del trabajo pueden ser:

- Ficheros de texto.
- Hojas de Excel.
- Correo electrónico.
- Etc.

Esta variada forma de recibir la "cartera de trabajo" supone numerosos problemas a las empresas de servicios tanto a la hora de distribuir el trabajo a sus empleados como de llevar un control en la realización de estos trabajos para la posterior comunicación y cobro de los servicios realizados. Distribuyendo y controlando las tareas a realizar de forma eficiente repercutirá en los costes de realización de dichas tareas, reduciendo dichos costes y aumentando el beneficio obtenido

Un ejemplo real de comunicación de la cartera de trabajo es mediante hojas Excel. La empresa de servicios recibe del ordenante un archivo con una hoja Excel donde se indican las tareas a realizar. Normalmente, se indica una tarea por cada fila incluida en el archivo Excel con la información necesaria para la realización de la tarea.

Otra forma de recibir las órdenes de trabajo es mediante correo electrónico. Esto supone un gran problema a las empresas de servicio a la hora de distribuir y controlar las tareas. Tanto si se recibe en un buzón genérico donde acceden varios empleados como si se recibe en los buzones particulares de cada uno de los empleados, la labor de control es complicada.

A continuación vemos una ilustración donde se puede observar de forma esquemática una situación real en el día a día muchas empresas de servicios

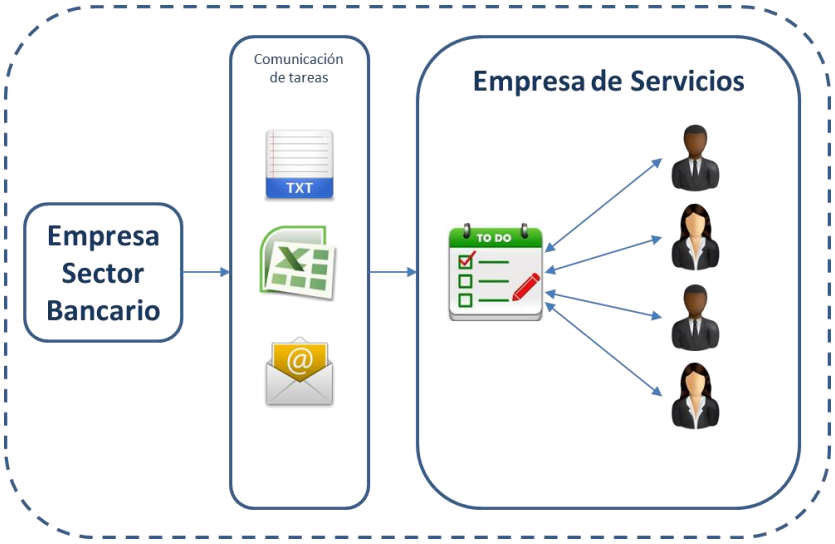

Figura 1. Esquema de ejemplo de la forma de comunicación de tareas entre empresas

En la ilustración podemos observar la forma de comunicar las tareas a realizar por parte de la empresa de servicios a través de ficheros de texto, hojas de Excel o correo electrónico. Una vez las tareas llegan a la empresa de servicios alimentan la lista de trabajos a realizar teniéndolos que distribuir entre los empleados que serán los que ejecuten las tareas necesarias para llevar a cabo la realización de dichas tareas.

2.1.1.1 Hoja Excel como medio de comunicación de tareas

Pondremos como ejemplo la llegada de una hoja de Excel como medio de comunicación de las tareas a realizar. La hoja Excel contiene la información de reclamaciones de tarjetas que debe realizar la empresa de servicios. La información se distribuye en el archivo de la siguiente forma:

- Cada fila es una reclamación a realizar, es decir una tarea a ejecutar
- Cada columna contiene la información necesaria para la realización de la tarea

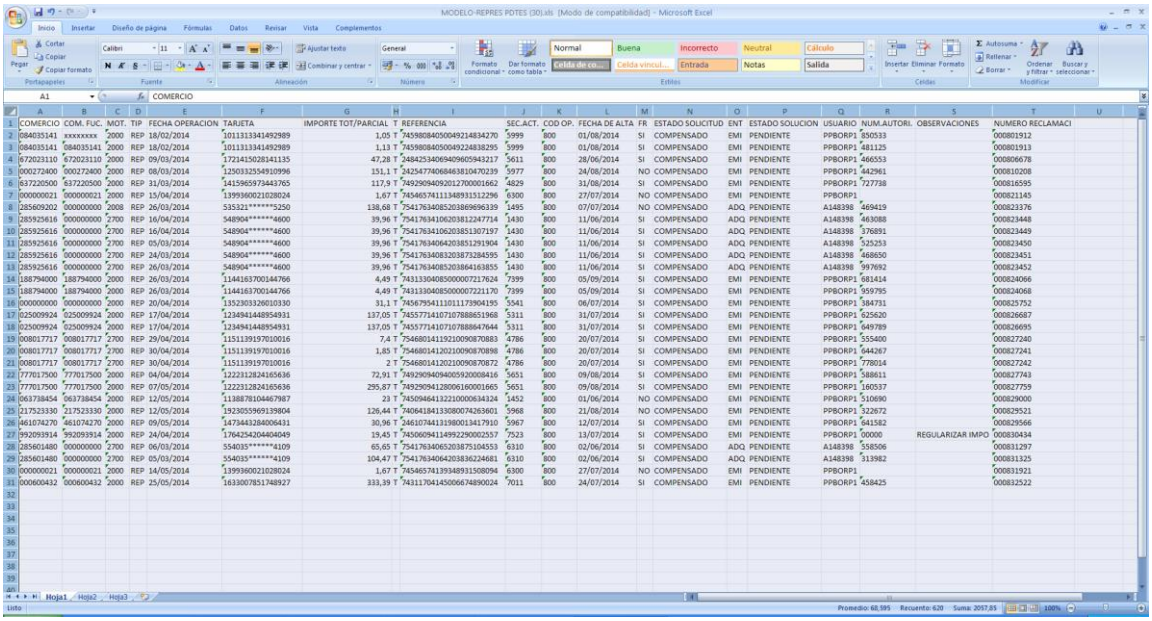

Figura 2. Ejemplo de Hoja Excel para la comunicación de tareas a realizar

Una vez llega la información a la empresa de servicios, esta debe distribuirla entre sus empleados para su realización y aquí es donde se encuentra con el mayor de los problemas. ¿Cómo distribuir el trabajo de forma eficiente?

Se plantean distintas opciones a la hora de la realización de las tareas y su distribución entre los empleados asignados a su ejecución:

- Se pueden asignar todas las tareas a un mismo empleado, de esta forma no habría ningún problema en la distribución. Por el contrario, el problema puede llegar en el volumen de las tareas a realizar. Si el volumen de tareas es elevado un solo empleado tardaría demasiado tiempo en la realización de todas las tareas

lo cual conllevaría a un incumplimiento del acuerdo de servicio pactado con la empresa origen[2].

- Otra opción es distribuir las tareas entre varios empleados de la empresa de servicios. Aquí se plantea la pregunta: ¿Cómo realizar esta distribución de manera eficiente? En el ejemplo vemos una hoja Excel con una serie de tareas a realizar. Pongamos el caso de que son, por ejemplo, tres empleados para realizar las tareas. La forma que se puede ocurrir de hacer el reparto de tareas sin la ayuda de ninguna herramienta, sería la siguiente:
    - Empleado 1: tareas de la fila 1 a la 10
    - Empleado 2: tareas de la fila 11 a la 21
    - Empleado 3: tareas de la fila 22 a la 31

Además se añade el problema de marcar como realizada la tarea. Se podría llegar al acuerdo que cada empleado que realice una tarea marque la fila de otro color. Esto implicaría acceder al fichero Excel de manera concurrente por parte de todos los empleados cosa que no es posible en ficheros de este tipo.

Observamos con el ejemplo presentado la necesidad de apoyo mediante una herramienta informática para la distribución de tareas entre los empleados y que sea lo suficientemente genérica para abarcar los distintos tipos de tareas a realizar y los distintos medios de comunicación de dichas tareas por parte de la empresa origen.

2.1.1.2 Mail como medio de comunicación de tareas

El mismo problema se plantearía en la empresa de servicios en el caso que las tareas se comunicaran mediante correo electrónico. En este caso se podrían dar los siguientes supuestos:

- Mail con el contenido de las tareas a realizar enviado al buzón particular del empleado encargado de realizar las tareas. En este caso el problema de distribución solo existiría en el caso de que las tareas a realizar fueran de un volumen elevado que requeriría la actuación de varios empleados. ¿Cómo distribuir y llevar un control de las tareas de forma eficiente? Si se reenvía el mail a todos los empleados implicados estaríamos perdiendo el control de la distribución de las tareas y de la realización de estas.

- Mail con el contenido de las tareas a realizar enviado a un buzón genérico el cual consultan todos los empleados encargados de realizar las tareas. En este caso el problema llegaría planteado en la forma de distribuir las tareas. Si todos los empleados implicados consultan el buzón ¿Cómo distribuir y llevar un control de las tareas de forma eficiente?

[2] Se denominará empresa origen a la empresa que externaliza un servicio para que la empresa contratada lo realice y comunique sus resultados

Tanto si la comunicación de las tareas a realizar llega a los buzones particulares de los empleados como si llega a un buzón genérico el cual consultan todos los empleados, nos encontramos un problema de llevar el control de las tareas realizadas.

Sin la ayuda una herramienta que se encargue de la distribución y control de los mails recibidos con las tareas a realizar, solo tendríamos la opción de que los empleados llegaran a un acuerdo para que cada uno de ellos se asignara la tarea, por ejemplo, moviendo el mail a una carpeta que solo él pudiera acceder. Esta forma no sería la más eficiente y conllevaría un serio problema a la hora de llevar un control de las tareas realizadas.

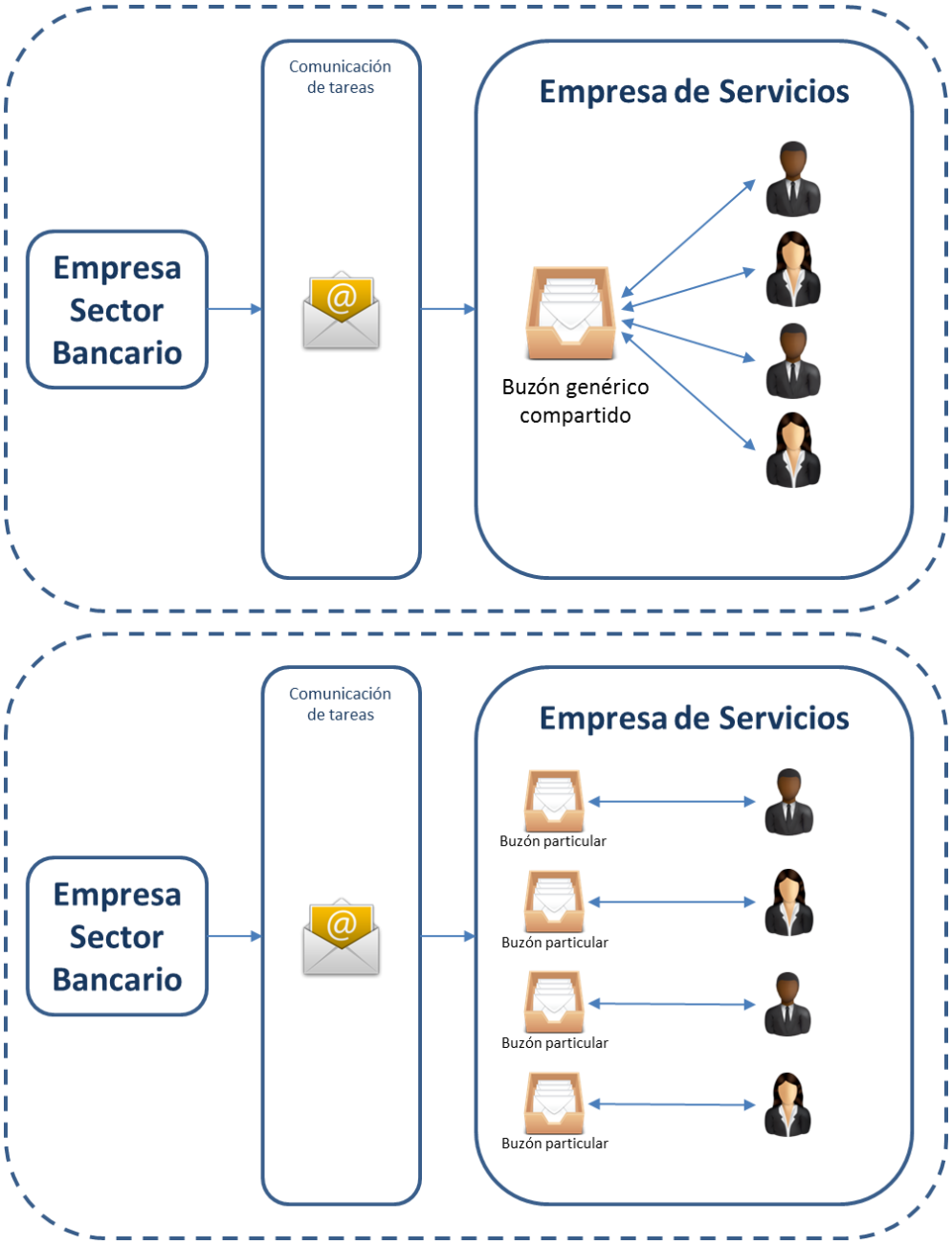

Figura 3. Ejemplo de correo electrónico para la comunicación de tareas a realizar

2.2 Análisis de soluciones existentes

Al analizar el estado actual del mercado en cuestión de aplicaciones dedicadas a la gestión y asignación de tareas observamos que existen un gran número de ellas. La mayoría de ellas están orientadas listas de tareas para tenerlas ordenadas y distribuidas en el entorno del grupo de trabajo.

A continuación se presentan una serie de ejemplos de aplicaciones de gestión y asignación de tareas existentes en el mercado actual [1]:

- Wunderlist

  Es una buena y sencilla aplicación que te permite crear listas de tareas y jerarquizar la importancia de cada una de ellas. Funciona fluidamente y ha alcanzado gran aceptación entre los usuarios. Además, te ofrece la posibilidad de delegar tareas, algo a tener en cuenta si se trata de un proyecto colaborativo. En lo que *Wunderlist* supera a los demás es en diseño: es muy limpio y te permite modificar el fondo a tu gusto.

- Nozbe

  Nozbe se nos muestra como una de las más completas: puedes personalizar tu cuenta con un logo personal, adjuntar archivos a cada proyecto, sincronizar con *Twitter, Dropbox, Evernote* y tu calendario de *Google*, enviar un correo al sistema para crear nuevas tareas, etc. La utilidad de esta herramienta la marca el usuario: puedes usar *Nozbe* como una simple lista de tareas o para proyectos más complejos que requieren la colaboración de los miembros de tu equipo.

- Teux Deux

  Esta es perfecta para quien configura su trabajo de semana en semana: el programa te separa en columnas la semana laboral y es muy fácil de usar. Destaca por su claridad y sencillez: lo único que ves en la pantalla son las tareas de los próximos días. Y es gratuito.

- Omnifocus

  Omnifocus merece aquí un buen puesto por destacar como el gestor GTD[3] más potente. Es muy popular y con él puedes marcar fechas límite a las tareas o proyectos y agregar mensajes de voz e imágenes. No es gratuito.

- Astrid

[3] Getting Things Done es un método de gestión de las actividades, se basa en el principio de que una persona necesita liberar su mente de las tareas pendientes guardándolas en un lugar específico. De este modo, no es necesario recordar lo que hay que hacer y se puede concentrar en realizar las tareas. [11]

Astrid se presenta con una imagen más clara, y las versiones para dispositivos móviles muestran también un buen diseño y son de fácil manejo: resulta muy sencillo crear una lista compartida y añadir colaboradores. Además, el sistema permite que tus ayudantes dejen comentarios a cada tarea, con lo que, de esta manera, podrán dar detalles de cómo han trabajado.

- Producteev

  Con Producteev podrás distribuir tus tareas en proyectos y añadir subtareas. La versión gratuita te permite invitar a una persona a tu workspace, pero con la suscripción mensual podrás añadir a más colaboradores e interactuar con ellos. Es aquí donde destaca esta herramienta, en la organización y distribución de las tareas de los miembros de tu equipo. Otro punto a favor de *Producteev* está en la posibilidad de adjuntar archivos y descargarlos en cualquier momento.

- Toodledo

  Con *Toodledo* puedes organizar las tareas según preferencias y prioridades. Si tienes una jornada de trabajo intensa pero no estás seguro de qué es lo que deberías hacer en primer lugar, la aplicación puede planificarlo por ti y sugerirte, a partir del tiempo estimado para cada una de las tareas, la fecha límite y la relevancia, cuál de ellas tienes que completar primero y cuál después. La versión gratuita incluye lo básico, pero si quieres además compartir tus listas y que la herramienta te ayude con la planificación de tu trabajo, entre otras funcionalidades, puedes suscribirte a la versión de pago.

2.2.1 Comparativa de las soluciones existentes

A continuación se muestra una comparativa de las soluciones expuestas en el apartado anterior en modo de tabla:

Solución	Gestión usuarios	Asignación	Reparto automático	Definición de campos
Wunderlist	☑	☑	☐	☐
Nozbe	☑	☑	☐	☐
Teux Deux	☑	☑	☐	☐
Omnifocus	☑	☑	☐	☐
Astrid	☑	☑	☐	☐
Producteev	☑	☑	☐	☐
Toodledo	☑	☑	☐	☐

Tabla 11. Comparativa de las soluciones existentes

2.3 Planteamiento de la solución

Una vez descritos los problemas a los que puede enfrentarse una empresa de servicios a la hora de afrontar la realización de las tareas encomendadas por la empresa origen, se tratará de plantear una solución lo más genérica y eficiente posible.

La finalidad de este proyecto es ofrecer una solución a los problemas descritos en el apartado anterior en forma de aplicación web. El nombre propuesto para esta aplicación es *GTareas*[4]. Esta aplicación web se encargará de gestionar las tareas llegadas de la entidad ordenante de trabajos (tareas encargadas desde empresas externas mediante una carga o tareas creadas desde la propia aplicación), distribuyéndolas y controlándolas de forma eficiente ofreciendo al usuario una serie de funcionalidades para facilitar la realización y el control del trabajo.

2.3.1 Aplicación web como solución

La decisión de que la solución sea la implementación de una aplicación web se puede apoyar en los siguientes motivos:

- Compatibilidad multiplataforma. Las aplicaciones web tienen un camino mucho más sencillo para la compatibilidad multiplataforma que las aplicaciones de software descargables. Varias tecnologías incluyendo Java, PHP, ASP y Ajax permiten un desarrollo efectivo de programas soportando todos los sistemas operativos principales.

- Actualización. Las aplicaciones basadas en web están siempre actualizadas con el último lanzamiento sin requerir que el usuario tome acciones pro-activas, y sin necesitar llamar la atención del usuario o interferir con sus hábitos de trabajo con la esperanza de que va a iniciar nuevas descargas y procedimientos de instalación.

- Inmediatez de acceso. Las aplicaciones basadas en web no necesitan ser descargadas, instaladas y configuradas. Las aplicaciones web están listas para trabajar sin importar cuál es su configuración o su hardware.

- Menos requerimientos de memoria. Las aplicaciones basadas en web tienen mucha menos demanda de memoria RAM de parte del usuario final que los programas instalados localmente. Al residir y correr en los servidores del proveedor, las aplicaciones basadas en web usan la memoria de las computadoras donde están ejecutando, sin reducir en el rendimiento de las computadoras cliente.

- Precio. Las aplicaciones basadas en web no requieren la infraestructura de distribución, soporte técnico y marketing requerido por el software descargable tradicional. Esto permite que las aplicaciones online cuesten una fracción de sus contrapartes descargables.

[4] Nombre propuesto para la aplicación web. GTareas, viene de Gestor de Tareas

- Múltiples usuarios concurrentes. Las aplicaciones basadas en web pueden realmente ser utilizadas por múltiples usuarios al mismo tiempo.

- Desarrollar aplicaciones en cualquier lenguaje. Una vez que las aplicaciones han sido separadas de computadoras locales y sistemas operativos específicos estas pueden ser escritas en prácticamente cualquier lenguaje de programación, mientras que para software escritorio se está limitado a usar el mismo lenguaje que el sistema operativo subyacente.

2.3.2 Módulos de la Aplicación web

La aplicación web a implementar se dividirá en una serie de módulos cuya funcionalidad se encargará de cada una de las partes necesarias para el cumplimiento de su objetivo.

- Administración de usuarios

  En este módulo se incluirá todo lo relacionado con la gestión de los usuarios que accederán a la aplicación. Incluirá la creación de cada uno de los usuarios y la modificación de sus datos, además de la gestión de las credenciales para su acceso al sistema.

- Administración de permisos

  Desde el módulo de administración de permisos se definirán los distintos perfiles que intervendrán en la aplicación así como los permisos que cada uno de estos perfiles dispondrán dentro del sistema. El módulo ofrecerá la funcionalidad de asignación de perfiles a los distintos usuarios, otorgando de esta forma los permisos correspondientes a cada uno de ellos.

- Administración de modelos de trabajo

  Este módulo será la base de la configuración de la aplicación. Desde este módulo se definirá la estructura de cada modelo de trabajo. Se definirán los campos de datos necesarios para la gestión de la tarea, los estados por los que podrá pasar una tarea, los usuarios que podrán trabajar en las tareas de este modelo. Además desde la administración de los modelos de trabajo se podrán cargar las tareas en las que los usuarios podrán trabajar

- Administración de colas de trabajo

  Desde este módulo se definirá la forma de trabajar configurando las distintas colas de trabajo en las que podrá trabajar un usuario de aplicación. Una cola de trabajo se define por una serie de criterios de búsqueda de tareas y una ordenación de estas para que cuando un usuario trabaje en la cola sistema le sirva tareas que cumplan los criterios de búsqueda y ordenación definidos.

- Gestión y distribución de tareas

  Este módulo es la base de la función de la aplicación: la gestión y distribución de tareas. Es desde este módulo donde los usuarios trabajarán con las tareas que le son asignadas tanto de manera manual por un administrador como automáticamente por el sistema desde las colas de trabajo definidas. Se podrán consultar todas las tareas creadas en el sistema y trabajar sobre ellas así como acceder a las distintas colas de trabajo creadas y trabajar sobre las tareas servidas por el sistema.

- Administración de buzones de correo

  En este módulo se configurarán los distintos buzones de correo con los que se desee trabajar. La idea es que estos buzones de correo sean buzones compartidos donde los usuarios accederían para surtirse de tareas. Desde este módulo se definirá la forma en la que cada mail se convertirá en una o varias tareas de un determinado modelo de trabajo a realizar por los usuarios. Esta definición se realizará a través de una serie de reglas definidas por los administradores de los buzones y los modelos de trabajo.

Figura 4. Representación de los módulos que componen la aplicación web

Capítulo 3

Desarrollo de la Aplicación GTareas

3.1 Introducción

En este capítulo se realizará una exposición del proceso de desarrollo seguido para implementar la aplicación GTareas, que se utilizará como herramienta para la gestión, reparto y asignación de tareas genéricas. La aplicación permitirá llevar a cabo una gestión completa y eficiente de las tareas introducidas en el sistema mediante diversos medios de comunicación.

Para llevar a cabo el desarrollo de la aplicación se utilizará la metodología de desarrollo ágil Scrum[5]. Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.

[5] Scrum es el nombre con el que se denomina a una de las metodologías de desarrollo ágiles existentes

Scrum también se utiliza para resolver situaciones en que no se está entregando al cliente lo que necesita, cuando las entregas se alargan demasiado, los costes se disparan o la calidad no es aceptable, cuando se necesita capacidad de reacción ante la competencia, cuando la moral de los equipos es baja y la rotación alta, cuando es necesario identificar y solucionar ineficiencias sistemáticamente o cuando se quiere trabajar utilizando un proceso especializado en el desarrollo de producto.[2]

La utilización de Scrum como metodología de desarrollo implicará la consecución de los siguientes pasos:

- Identificación de requisitos software.
- Formalización de requisitos software en el Product Backlog[6].
- Realización de cada uno de los Sprint[7], que llevarán asociados:
    - Aplicación de técnicas de análisis.
    - Aplicación de técnicas de diseño.
    - Codificación de funcionalidades.
    - Prueba de cada una de las funcionalidades.

La sucesión de los distintos Sprint permitirá mostrar un proceso iterativo e incremental donde el sistema GTareas alcanzará paulatinamente mayor número de funcionalidades hasta conseguir un producto acorde con la totalidad de requisitos establecidos.

3.2 Modelo de programación de la Aplicación web

La aplicación web se implementará siguiendo el modelo de programación por capas. Este modelo de programación tiene como objetivo separar la lógica de diseño de la lógica de negocios. Una de las ventajas que podemos destacar sobre este estilo es que el

[6] El product backlog o "lista de objetivos/requisitos priorizada" representa la visión y expectativas del cliente respecto a los objetivos y entregas del producto o proyecto. El cliente es el responsable de crear y gestionar la lista (con la ayuda del Facilitador y del equipo, quien proporciona el coste estimado de completar cada requisito). Dado que reflejar las expectativas del cliente, esta lista permite involucrarle en la dirección de los resultados del producto o proyecto. [6]

[7] Strint o ejecución de iteración, en Scrum un proyecto se ejecuta en bloques temporales cortos y fijos (iteraciones de un mes natural y hasta de dos semanas). Cada iteración tiene que proporcionar un resultado completo, un incremento de producto que sea potencialmente entregable, de manera que cuando el cliente (Product Owner) lo solicite sólo sea necesario un esfuerzo mínimo para que el producto esté disponible para ser utilizado. [7]

desarrollo del software se puede llevar a cabo en varios tipos de niveles, así, cuando suceda algún cambio solo nos iremos sobre el nivel requerido.

La programación por capas es una técnica de la ingeniería del software propia de la programación a objetos, que se divide en 3 capas: la capa de presentación o frontera, la capa de lógica de negocio y por último la capa de datos. [3]

- Capa de presentación
  Se refiere a la presentación del programa frente al usuario, esta presentación debe cumplir su propósito con el usuario final, una presentación fácil de usar y amigable. También las interfaces deben ser consistentes con la información dentro del software (Por ejemplo; en los formularios no debe haber más que lo necesario), tomar en cuenta los requerimientos del usuario, la capa de presentación va de la mano con capa de la lógica de negocio.

- Capa de lógica de negocio
  En esta capa es donde se encuentran los programas que son ejecutados, recibe las peticiones del usuario y posteriormente envía las respuestas tras el proceso. Esta capa es muy importantes pues es donde se establecen todas aquellas reglas que se tendrán que cumplir, decía anteriormente que la capa de presentación tiene comunicación con la capa de lógica de negocio ya que se tienen que comunicar para recibir las solicitudes y presentar los resultados.

- Capa de datos
  Esta capa es la que se encarga de hacer las transacciones con la base de datos y con otros sistemas para descargar o insertar información al sistema. La consistencia en los datos es sumamente importante, es decir, los datos que se ingresan o insertan deben ser precisos y consientes. Aquí definimos las consultas que vamos a realizar en la base de datos, o consultas para reporteo. La comunicación de esta capa con la capa de lógica de negocio se refiere a que la capa de datos es la que le enviara información a la capa de negocio para que sea procesada e ingresada en objetos según sea necesario (encapsulamiento). En esta capa usaremos los denominados *Objetos de Acceso a Datos (DAO)*. La ventaja de usar objetos de acceso a datos es que cualquier objeto de negocio (aquel que contiene detalles específicos de operación o aplicación) no requiere conocimiento directo del destino final de la información que manipula.[4]

La programación por capas nos proporciona una serie de ventajas a la hora de la implementación, ejemplo de estas ventajas serían:

- Reutilización de capas.
- Facilita la estandarización.
- Dependencias se limitan a intra-capa
- Contención de cambios a una o pocas capas

En este modelo, una aplicación se convierte en un conjunto de servicios de usuario, negocios y datos que satisface las necesidades de los procesos de negocios o procesa su soporte. Como los servicios están diseñados para el uso general y siguen lineamientos de interfaz publicados, pueden ser reutilizados y compartidos entre múltiples aplicaciones.[5]

Figura 5. Representación del modelo de programación por capas

3.3 Requisitos de la aplicación web

En los siguientes puntos se detallará la especificación de requisitos software que se ha tomado como base para implementar la aplicación GTareas. En el primer punto, se comentarán los fundamentos que se han empleado como referencia para la realización de la aplicación, mientras que en el segundo punto, se formalizará la especificación de requisitos elaborando el Product Backlog utilizado para dirigir la construcción del sistema.

3.3.1 Elaboración del Product Backlog

A continuación se expone la lista de requisitos de carácter general que se han establecido inicialmente. Esta lista aparece divida en tablas. En cada tabla se detallan cada uno de los requisitos que se han identificado. Cada uno de estos requisitos se define a través de los siguientes campos:

- ID: Identificador unívoco del requisito. Este identificador seguirá el siguiente patrón PB-N-XXX donde N identifica el Sprint en el que se ha establecido el requisito y XXX es un número de tres dígitos que identifica de forma unívoca el requisito.

- Nombre: nombre que identifica el requisito.

- Descripción: descripción asociada al requisito.

- Estabilidad: indica la posibilidad de que el requisito sea modificado por el Equipo de desarrollo a propuesta del Product Owner.

- Prioridad: prioridad establecida en el requisito, que determinará su incorporación en la aplicación GTareas. El rango de prioridades originalmente establecido es entre 0 y 1. Un valor mayor indica más prioridad.

- Coste: número utilizado para representar el coste asociado a la implementación del requisito. Se ha establecido un rango de coste entre 1 y 100. Un valor mayor indica más coste.

- Completado: porcentaje global del requisito que ha sido completado.

- Sprint 1: porcentaje del requisito completado en el Sprint 1.

- Sprint 2: porcentaje del requisito completado en el Sprint 2.

- Sprint 3: porcentaje del requisito completado en el Sprint 3.

- Sprint 4: porcentaje del requisito completado en el Sprint 4.

A continuación se expone la lista de requisitos de carácter general que se han establecido inicialmente:

PB-0-001							
Nombre		Interfaz web					
Descripción		La aplicación debe ser accesible para los usuarios utilizando únicamente un navegador. Dispondrá de una interfaz web que permitirá su utilización completa. La interfaz debe ser validada por los futuros usuarios de la aplicación.					
Estabilidad		☐Alta		☐Media		☒Baja	
Prioridad		0,9		Coste		70	
Completado		0%					
Sprint1	0%	Sprint2	0%	Sprint3	0%	Sprint4	0%

Tabla 12. Requisito PB-0-001.

PB-0-002							
Nombre		Multiusuario					
Descripción		La aplicación debe ser multiusuario, por lo que debe soportar la gestión completa de los mismos (altas, bajas y modificaciones).					
Estabilidad		☒Alta		☐Media		☐Baja	
Prioridad		0,8		Coste		25	
Completado		0%					
Sprint1	0%	Sprint2	0%	Sprint3	0%	Sprint4	0%

Tabla 13. Requisito PB-0-002.

PB-0-003			
Nombre	Perfiles de usuario		
Descripción	La aplicación dispondrá de varios perfiles. Se dispondrá de un módulo de administración de perfiles. Un perfil necesario es el de “Superadministrador” que configurará los parámetros de la aplicación y administrará los perfiles necesarios (en principio “Administrador” y “Usuario”). Cada uno de los perfiles creados llevará asignados los permisos necesarios.		
Estabilidad	☐Alta	☒Media	☐Baja
Prioridad	0,8	Coste	18

Completado		0%					
Sprint1	0%	Sprint2	0%	Sprint3	0%	Sprint4	0%

Tabla 14. Requisito PB-0-003.

PB-0-004							
Nombre		Gestión de modelos de trabajo					
Descripción		La aplicación dispondrá de un módulo de gestión de modelos de trabajo. El usuario con permisos podrá crear/modificar/eliminar modelos de trabajo.					
Estabilidad		☐Alta		☐Media		☒Baja	
Prioridad		0,9		Coste		90	
Completado		0%					
Sprint1	0%	Sprint2	0%	Sprint3	0%	Sprint4	0%

Tabla 15. Requisito PB-0-004.

PB-0-005							
Nombre		Gestión de campos dentro del modelo de trabajo					
Descripción		Cada modelo de trabajo dispondrá de la opción de crear/modificar/eliminar campos donde almacenar la información de la tarea. Además se indicará el tipo de dato de cada campo y una serie de características: ver en tarea, destacado, mostrar en resultado de búsqueda, mostrar en filtro de búsqueda, solo lectura, obligatorio.					
Estabilidad		☐Alta		☐Media		☒Baja	
Prioridad		0,9		Coste		25	
Completado		0%					
Sprint1	0%	Sprint2	0%	Sprint3	0%	Sprint4	0%

Tabla 16. Requisito PB-0-005.

PB-0-006	
Nombre	Gestión de estados dentro del modelo de trabajo
Descripción	Cada modelo de trabajo dispondrá de la opción de crear/modificar/eliminar estados por los que puede pasar cada una de las tareas y las relaciones entre ellos.

Estabilidad		☐Alta		☐Media		☒Baja	
Prioridad		0,9		Coste		25	
Completado		0%					
Sprint1	0%	Sprint2	0%	Sprint3	0%	Sprint4	0%

Tabla 17. Requisito PB-0-006.

PB-0-007							
Nombre		Gestión de permisos dentro del modelo de trabajo					
Descripción		La aplicación dispondrá de la posibilidad de asignar permisos dentro de cada modelo de trabajo. Dentro de la configuración del modelo se seleccionarán los usuarios que podrán trabajar con ese modelo y además los usuarios que tendrán permisos para administrar dicho modelo					
Estabilidad		☐Alta		☒Media		☐Baja	
Prioridad		0,7		Coste		15	
Completado		0%					
Sprint1	0%	Sprint2	0%	Sprint3	0%	Sprint4	0%

Tabla 18. Requisito PB-0-007.

PB-0-008							
Nombre		Carga de tareas					
Descripción		La aplicación dispondrá de la posibilidad de cargar masivamente las tareas desde un fichero Excel donde se indique el contenido de los campos definidos					
Estabilidad		☐Alta		☒Media		☐Alta	
Prioridad		0,5		Coste		15	
Completado		0%					
Sprint1	0%	Sprint2	0%	Sprint3	0%	Sprint4	0%

Tabla 19. Requisito PB-0-008.

PB-0-009	
Nombre	Administración de buzones de correo

Descripción		La aplicación dispondrá de la posibilidad de configurar el acceso a diversos buzones de correo. Además se podrá configurar la forma en que los correos entrantes se transforman en tareas a realizar					
Estabilidad		☐Alta		☒Media		☐Baja	
Prioridad		0,7		Coste		50	
Completado		0%					
Sprint1	0%	Sprint2	0%	Sprint3	0%	Sprint4	0%

Tabla 20. Requisito PB-0-009.

PB-0-010							
Nombre		Administración de colas de trabajo					
Descripción		El administrador de un modelo de trabajo tendrá la posibilidad de crear diversas colas de trabajo. Esto dará la posibilidad a los usuarios que realicen las tareas de conectarse a una cola y realizar las tareas que el sistema les vaya sirviendo. Las colas de trabajo consisten en una serie de criterios de filtrado y una serie de criterios de ordenación.					
Estabilidad		☐Alta		☒Media		☐Baja	
Prioridad		0,8		Coste		75	
Completado		0%					
Sprint1	0%	Sprint2	0%	Sprint3	0%	Sprint4	0%

Tabla 21. Requisito PB-0-010.

PB-0-011							
Nombre		Monitor de seguimiento					
Descripción		El administrador de un modelo de trabajo tendrá la posibilidad de realizar un seguimiento de las tareas a través de un monitor. Este monitor dará información sobre la evolución de las tareas dentro del sistema					
Estabilidad		☐Alta		☒Media		☐Baja	
Prioridad		0,3		Coste		20	
Completado		0%					
Sprint1	0%	Sprint2	0%	Sprint3	0%	Sprint4	0%

Tabla 22. Requisito PB-0-011.

PB-0-012							
Nombre		Consulta de tareas					
Descripción		La aplicación dispondrá de un buscador de tareas dentro del sistema. Un usuario con permisos tendrá la posibilidad de consultar las tareas de un determinado modelo por diversos filtros de búsqueda. Además podrá acceder a la información específica de cada tarea desde este buscador.					
Estabilidad		☐Alta		☐Media		☒Baja	
Prioridad		0,8		Coste		50	
Completado		0%					
Sprint1	0%	Sprint2	0%	Sprint3	0%	Sprint4	0%

Tabla 23. Requisito PB-0-012.

PB-0-013							
Nombre		Reparto de tareas					
Descripción		La aplicación dispondrá de un módulo de reparto de tareas. Cuando un usuario con permisos en alguna cola de trabajo acceda a este módulo el sistema automáticamente le servirá tareas para realizar.					
Estabilidad		☐Alta		☒Media		☐Baja	
Prioridad		0,9		Coste		70	
Completado		0%					
Sprint1	0%	Sprint2	0%	Sprint3	0%	Sprint4	0%

Tabla 24. Requisito PB-0-013.

PB-0-014			
Nombre	Asignación de tareas		
Descripción	La aplicación dispondrá de la posibilidad de asignación de tareas a un usuario en concreto. Cuando un usuario con permisos acceda a la gestión de una tarea podrá asignarla a cualquier otro usuario con permisos para que sea este último el único que pueda gestionarla		
Estabilidad	☐Alta	☒Media	☐Baja
Prioridad	0,8	Coste	75

Completado		0%					
Sprint1	0%	Sprint2	0%	Sprint3	0%	Sprint4	0%

Tabla 25. Requisito PB-0-013.

3.4 Análisis funcional de la aplicación web

3.4.1 Introducción al análisis funcional

El objetivo del análisis funcional es describir las funcionalidades del sistema mediante modelos o documentos de análisis. Identifica las interacciones con elementos externos y documenta las estructuras de información necesarias para completar el sistema.

- Su papel en el desarrollo de la aplicación es fundamental.
- Servirá de contrato con el cliente.
- Permitirá explicar qué funcionalidades tendremos que implementar.
- Nos permitirá estimar el esfuerzo que tendremos que realizar para obtener la solución.
- Podremos plantear qué pruebas tendremos que hacer para comprobar que lo que hemos hecho es lo que el cliente quiere.
- Cuando finalice el desarrollo servirá para garantizar que nuestros equipos de mantenimiento conozcan la funcionalidad de la aplicación, con lo que los evolutivos o correctivos no serán traumáticos.

Un aspecto muy importante y que no debemos olvidar es que aunque el Análisis Funcional es una fase dentro del ciclo de vida del desarrollo, el papel del analista no finaliza cuando acaba la fase de análisis y entrega los artefactos de análisis que ha realizado (p.ej.: el documento de Análisis Funcional). Todo lo contrario, a partir de ese momento su objetivo debe ser que todas las personas involucradas en el desarrollo entiendan e implementen las funcionalidades planteadas.[6]

En este proyecto se utilizará uno de los métodos más extendidos en la ingeniería del software para realizar el análisis funcional, los *Casos de Uso*[8].

[8] El modelado de casos de uso es un método orientado a los usuarios para identificar necesidades funcionales de un nuevo sistema de información. El modelado de casos de uso es una técnica que permite modelar las funciones de un sistema en términos de eventos, de quien inicia los eventos y de cómo responde el sistema a esos eventos [9]

3.4.2 Análisis del requisito PB-0-001

Según el contenido del *Product Backlog* elaborado en el punto 3.3.1, el requisito identificado como PB-0-001 indica lo siguiente:

> La aplicación debe ser accesible para los usuarios utilizando únicamente un navegador. Dispondrá de una interfaz web que permitirá su utilización completa. La interfaz debe ser validada por los futuros usuarios de la aplicación.

Este requisito establece la necesidad de que la aplicación sea utilizable por cualquier usuario disponiendo únicamente de un navegador Web. A priori, se considera que es un requisito de baja estabilidad puesto que la propia interfaz debe ser validada por los usuarios finales de la aplicación. Tras plantear este requisito se determina que debe utilizarse una arquitectura cliente-servidor en la que múltiples usuarios podrán utilizar de forma simultánea el sistema y que adicionalmente, permitirá una gestión centralizada de la información. Se trata de un requisito transversal puesto que afecta al resto de componentes de la aplicación.

Para cumplir con las necesidades establecidas en el requisito PB-0-001, será necesario realizar las siguientes tareas:

- Selección de la arquitectura (framework, servidor de aplicaciones, herramientas de control y software subyacente).
- Configuración y parametrización de aplicación.
- Diseño de la capa de presentación de la aplicación.
- Elaboración de un prototipo para realizar su validación.

3.4.2.1 Arquitectura de la aplicación web

Como ya se ha explicado en el punto 3.2 “Modelo de programación de la Aplicación web”, el modelo de desarrollo de software utilizado es el modelo de programación por capas. Este modelo de programación tiene separa la lógica de diseño de la lógica de negocio. Las capas en las que se divide son: la capa de presentación o frontera, la capa de lógica de negocio y por último la capa de datos.

En cuanto a la arquitectura de software, a continuación se exponen los puntos en los que se indica cada uno de los componentes que han sido necesarios para el desarrollo de la aplicación web:

- Sistema operativo: se ha seleccionado el sistema operativo de Microsoft Windows 7 debido a su gran difusión y aceptación por parte de los usuarios. Esta plataforma se utilizará como base para la instalación del servidor de aplicaciones y para la ejecución de los clientes asociados. Versión: Windows 7 Ultimate.

- SGBD[9]: el sistema gestor de base de datos instalado es MySql 5.6. En el propio SGBD se creará un esquema utilizado por la aplicación GTareas.

- Para la implementación de la aplicación se utilizará el lenguaje JAVA, debido a sus características intrínsecas como la portabilidad, sencillez, robustez y seguridad. Versión: JDK 1.7.0.

- Como servidor de aplicaciones se empleará Apache Tomcat. Este servidor de aplicaciones se utilizará como servidor Web y como contenedor de los componentes ejecutables vía Web (Servlets). Versión: Apache Tomcat 7.0.39.

- El framework utilizado como base para la construcción de la aplicación será Spring. Se ha escogido este framework debido al gran número de extensiones y características que proporciona para construir aplicaciones web. Versión: Spring 3.1.2.

- La herramienta software utilizada para la gestión y construcción del proyecto ha sido Apache Ant. Esta herramienta se utiliza para describir el proyecto software a construir, estableciendo las dependencias entre módulos, componentes externos y el orden de construcción de los elementos. Versión: Apache Ant 1.9.2.

- El sistema de control de versiones seleccionado para el proyecto es Subversión. Un sistema de control de versiones facilita en gran medida el desarrollo colaborativo, distribuido y permite la recuperación de versiones anteriores cuando así se requiera. Versión: Subversion 1.6.

3.4.2.2 Configuración del servidor de aplicaciones

Como paso previo al desarrollo de la aplicación GTareas se procede a establecer la configuración del servidor de aplicaciones Apache Tomcat. Tras establecer esta configuración, se lleva a cabo la incorporación de todos los elementos asociados al desarrollo (packages, librerías…) utilizando Apache Ant. Después de llevar a cabo esta acción, se determina la estructura de directorios que se utilizará para albergar el código fuente de la aplicación. Finalmente, se comienza a definir el contenido de los ficheros XML encargados de soportar la configuración de la aplicación.

En la siguiente captura de pantalla se expone la configuración del servidor Apache Tomcat:

[9] Sistema gestor de base de datos

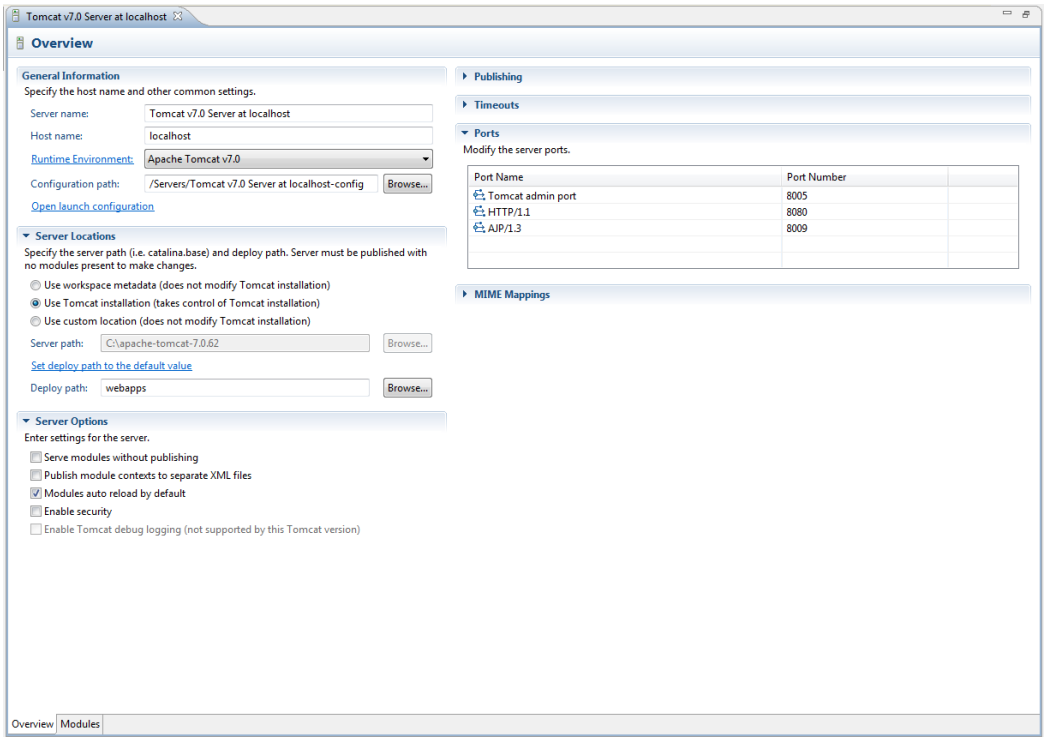

Figura 6. Configuración del servidor de aplicaciones Apache Tomcat

En la siguiente ilustración se muestra el fichero de configuración relativo al conjunto de conexiones JDBC[10] utilizado para el acceso directo a la base de datos:

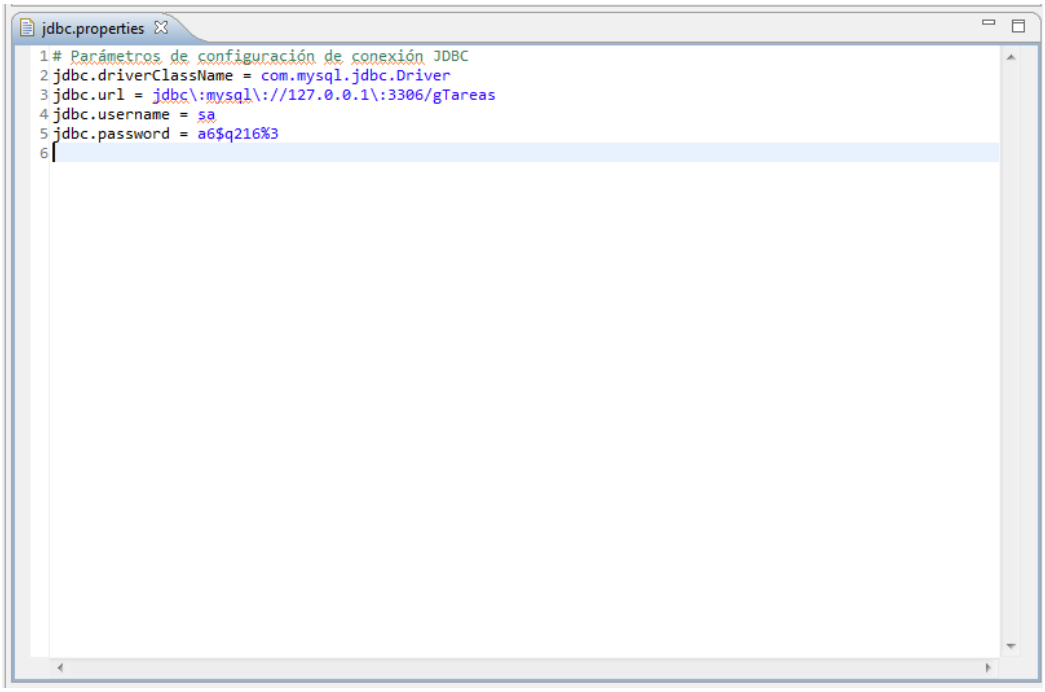

Figura 7. Archivo de configuración para la conexión con la base de datos

[10] Java Data Base Connectivity, es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java [12]

3.4.2.3 Diseño de la capa de presentación de la aplicación web

Inicialmente se elabora un diseño de la pantalla principal que utilizará la aplicación GTareas. Este diseño se utilizará como base para la definición del aspecto final de la aplicación:

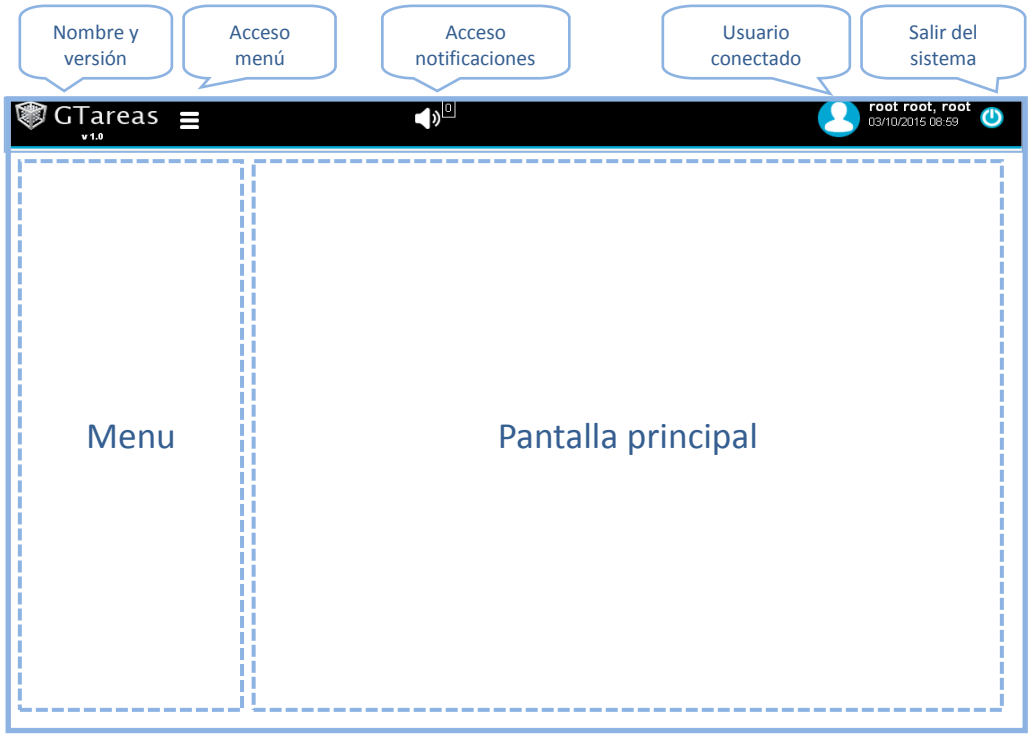

Figura 8. Plantilla del diseño de las pantallas de la aplicación web

En la ilustración anterior podemos observar la plantilla que encapsula el diseño de las pantallas de la aplicación web y podemos observar los siguientes detalles:
En la parte superior izquierda aparecerá el identificador de la aplicación y el enlace para acceder al menú de la aplicación.

En la parte superior derecha aparecerá el usuario, identificador de conexión y el perfil validado en la aplicación. En la parte superior central aparece el enlace a la ventana de notificaciones del usuario. Aquí se mostrará el número de notificaciones que el usuario tiene pendientes de leer. Debajo de los elementos anteriores se presenta la pantalla principal de la aplicación donde se mostrarán todos los elementos correspondientes al módulo al que se acceda.

Después de establecer el diseño inicial de la pantalla, se elabora un prototipo básico que permite la navegación entre dos pantallas de la aplicación. Este prototipo se utilizará como base para realizar la presentación final de la aplicación y simultáneamente, permitirá la validación de la misma.

3.4.3 Análisis del requisito PB-0-002

Según el contenido del *Product Backlog* elaborado en el punto 3.3.1, el requisito identificado como PB-0-002 indica lo siguiente:

La aplicación debe ser multiusuario, por lo que debe soportar la gestión completa de los mismos (altas, bajas y modificaciones).

La aplicación debe ser accesible por múltiples usuarios. Por esta razón, la aplicación debe contener un módulo de administración de usuarios. Se posibilitarán las funcionalidades de alta, modificación y baja de usuarios.

Para plasmar el comportamiento deseado de la aplicación web, se usarán los *diagramas de casos de uso*. Los diagramas de casos de uso describen las relaciones y las dependencias entre un grupo de casos de uso y los actores participantes en el proceso. Es importante resaltar que los diagramas de casos de uso no están pensados para representar el diseño y no pueden describir los elementos internos de un sistema. Los diagramas de casos de uso sirven para facilitar la comunicación con los futuros usuarios del sistema, y con el cliente, y resultan especialmente útiles para determinar las características necesarias que tendrá el sistema.[7]

3.4.3.1 Diagrama de casos de uso

En la siguiente figura aparece el diagrama de casos de uso que permite representar la gestión de usuarios que se llevará a cabo en la aplicación:

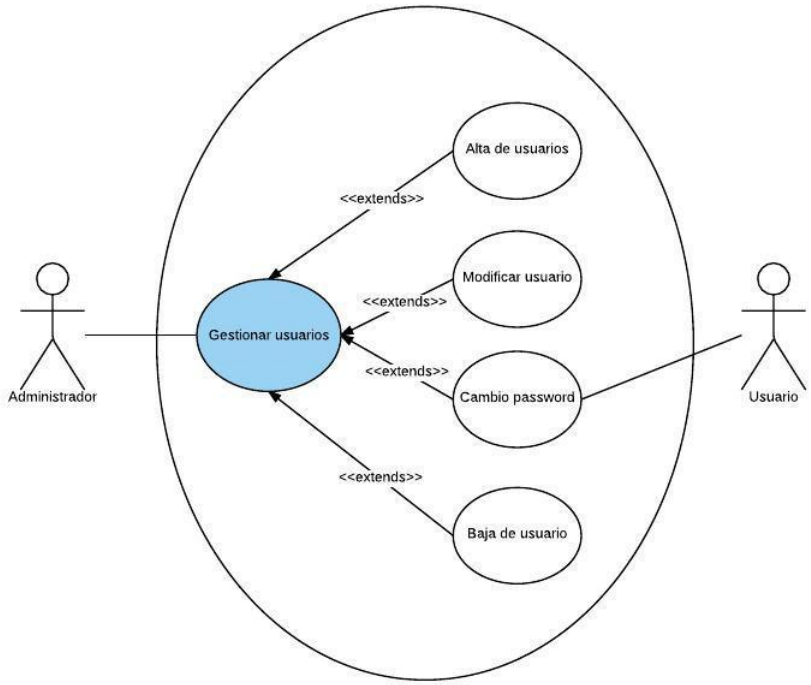

Figura 9. Diagrama de casos de uso "Gestionar Usuarios"

En este diagrama de casos de uso podemos observar como se ha representado el rol de "Administrador" y el rol de "Usuario". En el detalle de esta representación podemos ver que aquellos usuarios que dispongan del rol "Administrador" podrán llevar a cabo todas las acciones asociadas a la gestión de usuarios (altas, bajas y modificaciones). Un usuario que disponga del rol "Usuario" únicamente podrá llevar a cabo la modificación de su propia contraseña de acceso al sistema.

3.4.3.2 Tareas asociadas al requisito PB-0-002

Una vez analizado el requisito PB-0-002 que indica que a la aplicación web estará habilitada a múltiples usuarios y que indica la necesidad de una administración de usuario, se describirán las tareas a realizar en el desarrollo e implementación para el cumplimiento de este requisito:

- Diseño de esquema de base de datos con las tablas necesarias para el registro de usuarios
- Capa de acceso a base de datos tanto para el alta, la modificación y la baja de los usuarios así como la asignación de perfiles
- Servicio de la capa de negocio que se encargará tanto de la administración como de la autenticación de usuarios. Gestionará tanto la creación y mantenimiento de los usuarios como la comprobación de las credenciales de identificación de los usuarios que intenten acceder al sistema.
- Diseño de pantalla de administración de usuarios.
- Diseño de pantalla de login de acceso de usuarios.

3.4.4 Análisis del requisito PB-0-003

Según el contenido del *Product Backlog* elaborado en el punto 3.3.1, el requisito identificado como PB-0-003 indica lo siguiente:

> La aplicación dispondrá de varios perfiles. Se dispondrá de un módulo de administración de perfiles. Un perfil necesario es el de "Superadministrador" que configurará los parámetros de la aplicación y administrará los perfiles necesarios (en principio "Administrador" y "Usuario"). Cada uno de los perfiles creados llevará asignados los permisos necesarios.

La aplicación debe disponer de perfiles para poder asignar permisos a los distintos usuarios que accederán a la aplicación. Por esta razón, la aplicación debe contener un módulo de administración de perfiles solo accesible por el usuario "Superadministrador". Este usuario podrá crear todos los permisos que considere oportunos, cada uno de ellos con una serie de permisos sobre las funcionalidades de la aplicación. Una vez creados los perfiles, estos estarán disponibles para asignarlos a los usuarios de la aplicación.

3.4.4.1 Diagrama de casos de uso

En la siguiente figura aparece el diagrama de casos de uso que permite representar la administración de perfiles que se llevará a cabo en la aplicación:

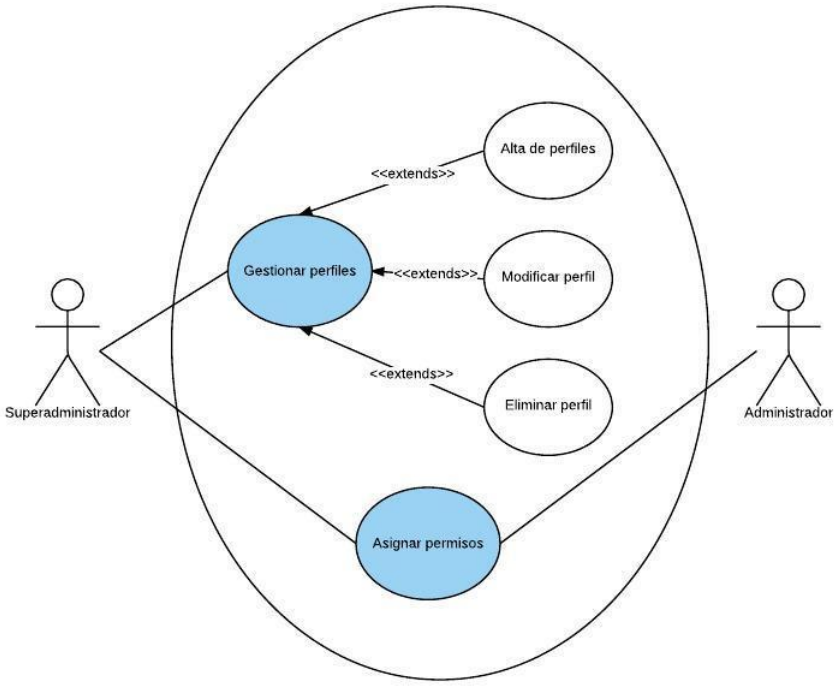

Figura 10. Diagrama de casos de uso "Gestionar Perfiles"

En este diagrama de casos de uso podemos observar como se ha representado el rol de "Superadministrador". En el detalle de esta representación podemos ver que aquellos usuarios que dispongan del rol "Superadministrador" (administrador del sistema) podrán llevar a cabo todas las acciones asociadas a la gestión de perfiles (altas, bajas y modificaciones). Además un usuario "Administrador" podrá asignar los perfiles ya creados a los usuarios de la aplicación.

3.4.4.2 Tareas asociadas al requisito PB-0-003

Una vez analizado el requisito PB-0-003 que indica que a la aplicación web debe disponer de perfiles para poder asignar permisos a los distintos usuarios que accederán a la aplicación, se describirán las tareas a realizar en el desarrollo e implementación para el cumplimiento de este requisito:

- Diseño de esquema de base de datos con las tablas necesarias para el registro de perfiles.

- Capa de acceso a base de datos tanto para el alta, la modificación y la baja de los perfiles.

- Servicio de la capa de negocio que se encargará de la administración de los perfiles que estarán disponibles para su asignación a los usuarios
- Diseño de pantalla de administración de perfiles
- Diseño de pantalla de asignación de perfiles a los usuarios

3.4.5 Análisis del requisito PB-0-004 al PB-0-007

Según el contenido del Product Backlog elaborado en el punto 3.3.1, el requisito identificado como PB-0-004 indica lo siguiente:

> La aplicación dispondrá de un módulo de gestión de modelos de trabajo. El usuario con permisos podrá crear/modificar/eliminar modelos de trabajo.

Según el contenido del Product Backlog elaborado en el punto 3.3.1, el requisito identificado como PB-0-005 indica lo siguiente:

> Cada modelo de trabajo dispondrá de la opción de crear/modificar/eliminar campos donde almacenar la información de la tarea. Además se indicará el tipo de dato de cada campo y una serie de características: ver en tarea, destacado, mostrar en resultado de búsqueda, mostrar en filtro de búsqueda, solo lectura, obligatorio.

Según el contenido del Product Backlog elaborado en el punto 3.3.1, el requisito identificado como PB-0-006 indica lo siguiente

> Cada modelo de trabajo dispondrá de la opción de crear/modificar/eliminar estados por los que puede pasar cada una de las tareas y las relaciones entre ellos.

Según el contenido del Product Backlog elaborado en el punto 3.3.1, el requisito identificado como PB-0-007 indica lo siguiente

> La aplicación dispondrá de la posibilidad de asignar permisos dentro de cada modelo de trabajo. Dentro de la configuración del modelo se seleccionarán los usuarios que podrán trabajar con ese modelo y además los usuarios que tendrán permisos para administrar dicho modelos.

3.4.5.1 Diagrama de casos de uso

En la siguiente figura aparece el diagrama de casos de uso que permite representar la administración de modelos de trabajo que se llevará a cabo en la aplicación:

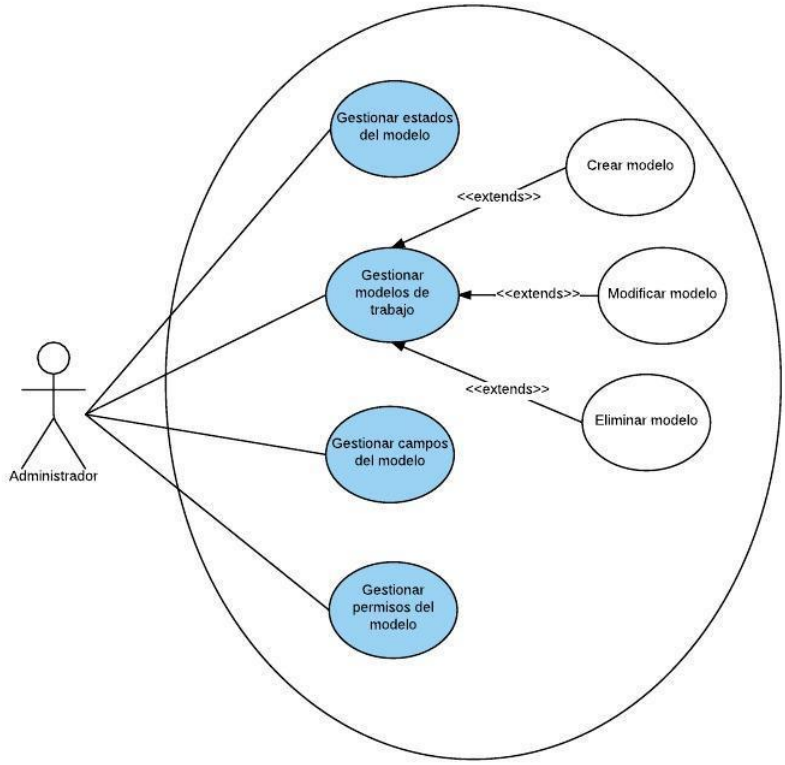

Figura 11. Diagrama de casos de uso "Gestionar Modelos de trabajo"

En el detalle de este diagrama se representa a los usuarios que dispongan del rol "Administrador". Estos usuarios podrán llevar a cabo todas las acciones asociadas a la gestión de modelos de trabajo (altas, bajas y modificaciones).

3.4.5.2 Tareas asociadas a los requisitos PB-0-004 al PB-0-007

Una vez analizado los requisitos PB-0-004 al PB-0-007, se describirán las tareas a realizar en el desarrollo e implementación para el cumplimiento de estos requisitos:

- Diseño de esquema de base de datos con las tablas necesarias para el registro de modelos de trabajo, además del registro de campos, estados y permisos para estos modelos de trabajo

- Capa de acceso a base de datos tanto para el alta, la modificación y la baja de los modelos de trabajo, así como para el registro de los estados, campos y permisos.

- Servicio de la capa de negocio que se encargará de la administración de los modelos de trabajo, sus estados, sus campos y sus permisos.

- Diseño de pantalla de administración de modelos de trabajo.

- Diseño de pantalla para el mantenimiento de estados por los que podrán pasar las tareas que pertenezcan al modelo de trabajo.

- Diseño de pantalla para el mantenimiento de campos donde se guardará la información necesaria de las tareas que pertenezcan al modelo de trabajo.

- Diseño de pantalla para el mantenimiento de los usuarios con permisos para gestionar las tareas que pertenezcan al modelo de trabajo.

3.4.6 Análisis del requisito PB-0-008

Según el contenido del Product Backlog elaborado en el punto 3.3.1, el requisito identificado como PB-0-008 indica lo siguiente:

La aplicación dispondrá de la posibilidad de cargar masivamente las tareas desde un fichero Excel donde se indique el contenido de los campos definidos.

Las tareas de un modelo de trabajo podrán ser cargadas en el sistema desde un fichero Excel que contendrá una tarea por fila y en cada columna los campos con la información necesaria para la gestión de la tarea.

3.4.6.1 Diagrama de casos de uso

En la siguiente figura aparece el diagrama de casos de uso que permite representar la carga de tareas desde un fichero Excel que se llevará a cabo en la aplicación:

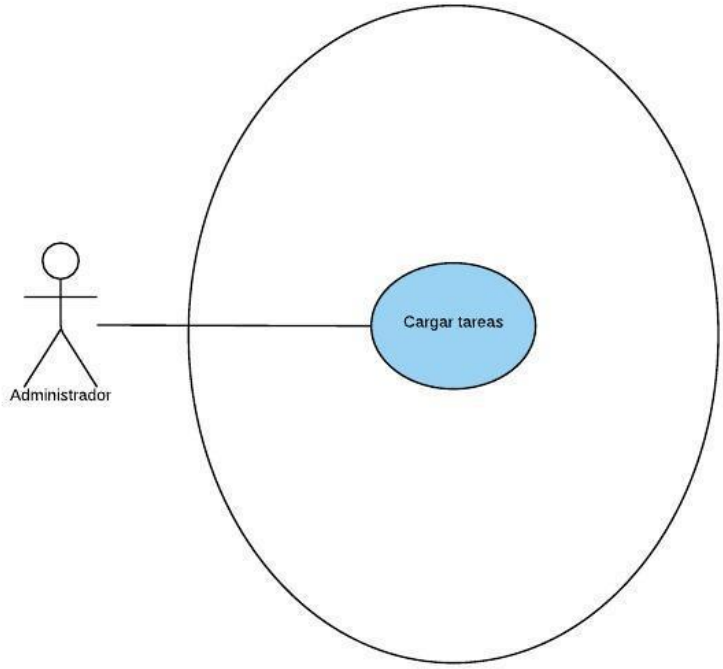

Figura 12. Diagrama de casos de uso "Cargar tareas"

3.4.6.2 Tareas asociadas a los requisitos PB-0-008

Una vez analizado los requisitos PB-0-008, se describirán las tareas a realizar en el desarrollo e implementación para el cumplimiento de estos requisitos:

- Diseño de esquema de base de datos con las tablas necesarias para el registro de tareas de un modelo de trabajo.
- Servicio de la capa de negocio que se encargará de la lectura del fichero Excel y la adaptación de los datos recibidos.
- Capa de acceso a base de datos para el registro de las tareas.
- Diseño de pantalla para la carga de tareas desde un fichero Excel.

3.4.7 Análisis del requisito PB-0-009

Según el contenido del Product Backlog elaborado en el punto 3.3.1, el requisito identificado como PB-0-009 indica lo siguiente:

La aplicación dispondrá de la posibilidad de configurar el acceso a diversos buzones de correo. Además se podrá configurar la forma en que los correos entrantes se transforman en tareas a realizar.

Otro método de entrada de tareas al sistema será mediante los correos recibidos en buzones de correo configurados para ello. Se registrarán los datos de conexión a los buzones y el sistema accederá a los correos para registrarlos como tareas mediante reglas predefinidas.

3.4.7.1 Diagrama de casos de uso

En la siguiente figura aparece el diagrama de casos de uso que permite representar la administración de buzones de correo:

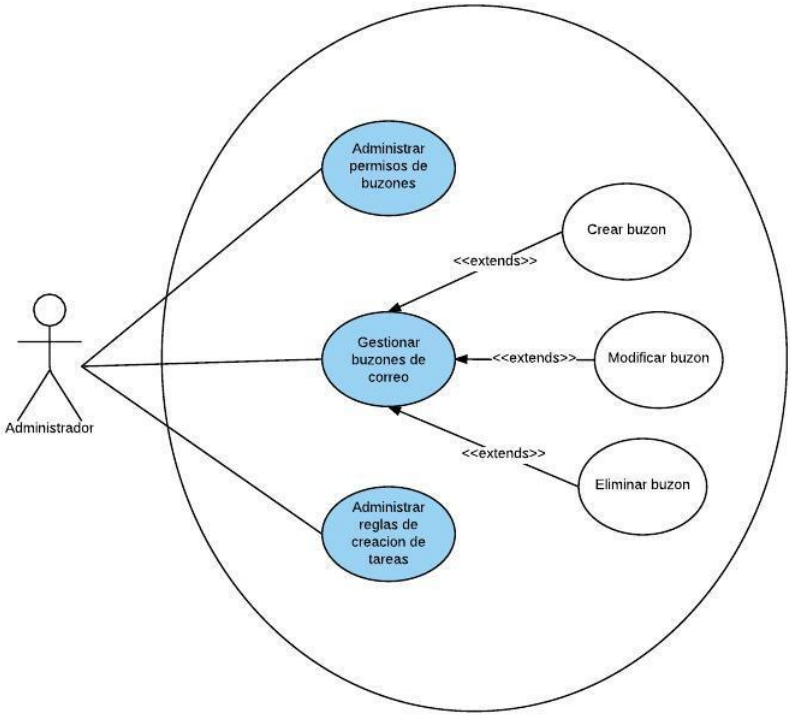

Figura 13. Diagrama de casos de uso "Gestionar buzones de correo"

3.4.7.2 Tareas asociadas a los requisitos PB-0-009

Una vez analizado los requisitos PB-0-009, se describirán las tareas a realizar en el desarrollo e implementación para el cumplimiento de estos requisitos:

- Diseño de esquema de base de datos con las tablas necesarias para el registro de la configuración de acceso a buzones de correo.
- Capa de acceso a base de datos para el registro de buzones, tareas y permisos.
- Servicio de la capa de negocio que se encargará de la administración de los buzones, registro de tareas y administración de permisos.
- Diseño de pantalla para la administración de buzones de correo. Además de la posibilidad de crear reglas de registro de tareas a partir de los correos y la posibilidad de administrar los permisos de los buzones.

3.4.8 Análisis del requisito PB-0-010

Según el contenido del Product Backlog elaborado en el punto 3.3.1, el requisito identificado como PB-0-010 indica lo siguiente:

El administrador de un modelo de trabajo tendrá la posibilidad de crear diversas colas de trabajo. Esto dará la posibilidad a los usuarios que realicen las tareas de conectarse a una cola y realizar las tareas que el sistema les vaya sirviendo. Las colas de

trabajo consisten en una serie de criterios de filtrado y una serie de criterios de ordenación.

En la aplicación web existirá la posibilidad de crear colas de trabajo. Estas colas de trabajo serán, básicamente, un listado de tareas resultantes de una búsqueda realizada mediante unos criterios de filtrado y unos criterios de ordenación definidos. Para estas colas de trabajo se definirán los permisos de acceso necesarios. Una vez creadas las colas de trabajo deseadas, los usuarios con permisos podrán conectarse a estas colas y comenzar a trabajar con las tareas que el sistema les sirva.

3.4.8.1 Diagrama de casos de uso

En la siguiente figura aparece el diagrama de casos de uso que permite representar la administración de colas de trabajo:

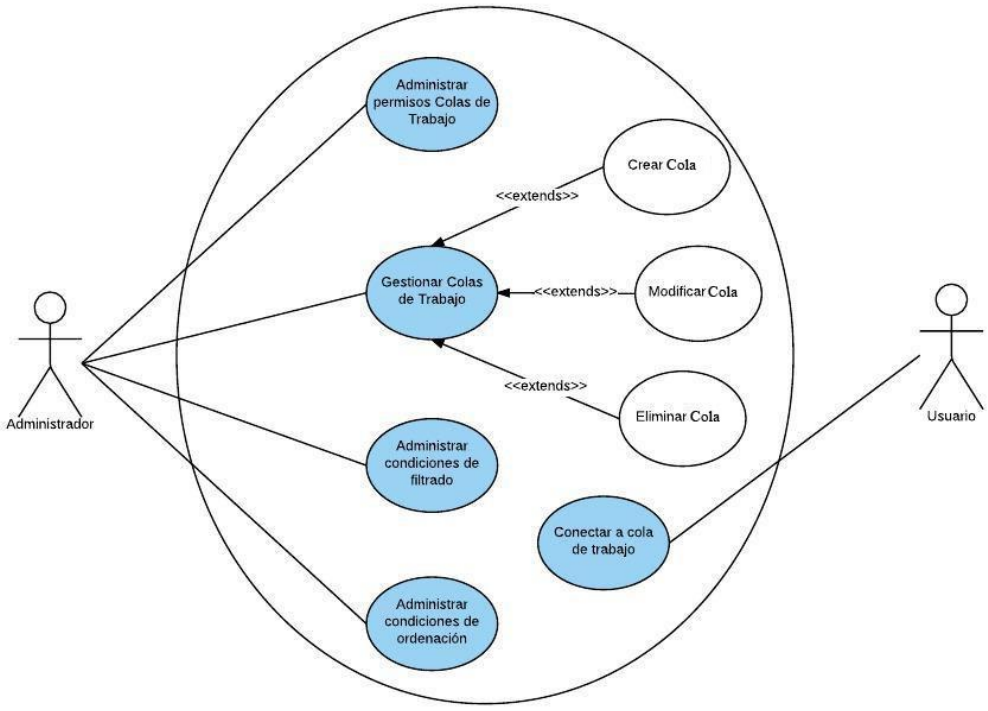

Figura 14. Diagrama de casos de uso "Gestionar colas de trabajo"

Observamos en el diagrama como entran en acción los usuarios tanto con perfil "Administrador" como los usuarios con perfil "Usuario". Los administradores gestionarán y configurarán las colas de trabajo así como establecerán los permisos otorgando acceso a los usuarios pertinentes. Los usuarios con perfil "Usuario", en caso de tener permisos, podrán conectarse a la cola de trabajo para que el sistema comience a servirles tareas a realizar.

3.4.8.2 Tareas asociadas a los requisitos PB-0-010

Una vez analizado los requisitos PB-0-010, se describirán las tareas a realizar en el desarrollo e implementación para el cumplimiento de estos requisitos:

- Diseño de esquema de base de datos con las tablas necesarias para el registro de colas de trabajo y almacenar criterios de filtrado y ordenación.
- Capa de acceso a base de datos para el registro de los datos de las colas de trabajo y sus criterios de filtrado y ordenación.
- Servicio de la capa de negocio que se encargará de la administración de las colas de trabajo y de gestionar y tratar sus criterios de filtrado y ordenación.
- Diseño de pantalla para la administración de colas de trabajo. Además de la posibilidad de configurar las colas mediante criterios de filtrado y criterios de ordenación.

3.4.9 Análisis del requisito PB-0-011

Según el contenido del Product Backlog elaborado en el punto 3.3.1, el requisito identificado como PB-0-011 indica lo siguiente:

El administrador de un modelo de trabajo tendrá la posibilidad de realizar un seguimiento de las tareas a través de un monitor. Este monitor dará información sobre la evolución de las tareas dentro del sistema.

En la aplicación web existirá la posibilidad de monitorizar la evolución de la realización de tareas en las distintas colas de trabajo configuradas.

3.4.9.1 Diagrama de casos de uso

En la siguiente figura aparece el diagrama de casos de uso que permite representar la administración de colas de trabajo:

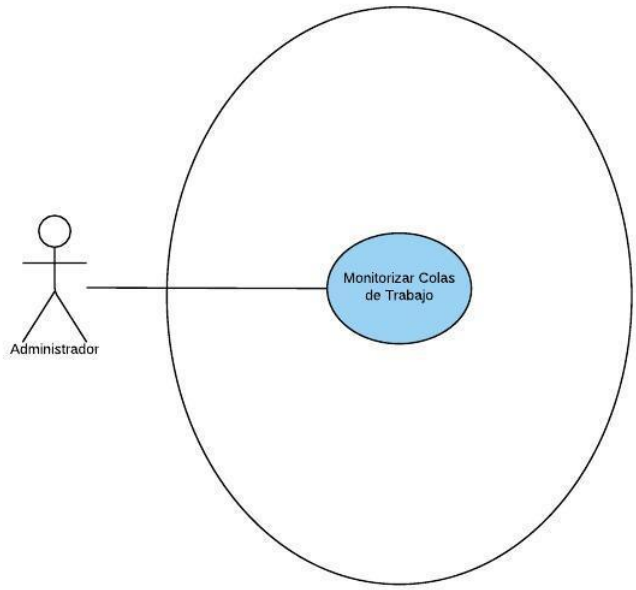

Figura 15. Diagrama de casos de uso "Monitorizar colas de trabajo"

3.4.9.2 Tareas asociadas a los requisitos PB-0-011

Una vez analizado los requisitos PB-0-011, se describirán las tareas a realizar en el desarrollo e implementación para el cumplimiento de estos requisitos:

- Capa de acceso a base de datos para la consulta de tareas dentro de las colas de trabajo de un modelo de trabajo.
- Servicio de la capa de negocio que se encargará de recoger la información necesaria para el seguimiento de tareas en las colas de trabajo.
- Diseño de pantalla para la monitorización de las colas de trabajo dentro de un modelo de trabajo.

3.4.10 Análisis del requisito PB-0-012

Según el contenido del Product Backlog elaborado en el punto 3.3.1, el requisito identificado como PB-0-012 indica lo siguiente:

La aplicación dispondrá de un buscador de tareas dentro del sistema. Un usuario con permisos tendrá la posibilidad de consultar las tareas de un determinado modelo por diversos filtros de búsqueda. Además podrá acceder a la información específica de cada tarea desde este buscador.

En la aplicación web existirá la posibilidad de realizar búsquedas de tareas mediante diversos criterios de filtrado y de visualizar el resultado atendiendo a criterios de ordenación definidos. Se posibilitará la opción de acceder a las tareas mostradas en los resultados de búsqueda.

3.4.10.1 Diagrama de casos de uso

En la siguiente figura aparece el diagrama de casos de uso que permite representar la administración de colas de trabajo:

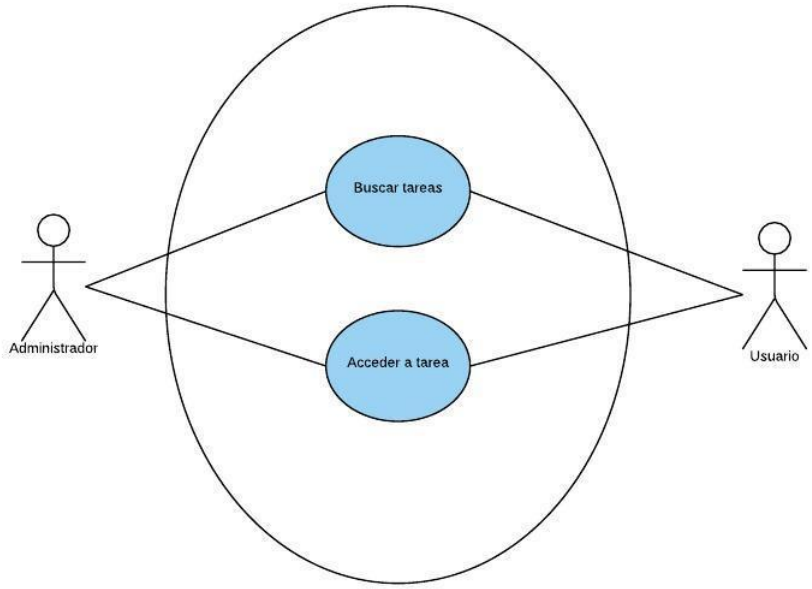

Figura 16. Diagrama de casos de uso "Buscador de Tareas"

3.4.10.2 Tareas asociadas a los requisitos PB-0-012

Una vez analizado los requisitos PB-0-012, se describirán las tareas a realizar en el desarrollo e implementación para el cumplimiento de estos requisitos:

- Capa de acceso a base de datos para la consulta de tareas y acceso a sus datos.
- Servicio de la capa de negocio que se encargará de recoger los resultados de las búsquedas con los criterios definidos.
- Diseño de pantalla para la búsqueda de tareas mediante criterios de filtrado y mostrar resultados siguiendo criterios de ordenación. Posibilidad de acceso a los datos de las tareas resultantes.

3.4.11 Análisis del requisito PB-0-013

Según el contenido del Product Backlog elaborado en el punto 3.3.1, el requisito identificado como PB-0-013 indica lo siguiente:

La aplicación dispondrá de un módulo de reparto de tareas. Cuando un usuario con permisos en alguna cola de trabajo acceda a este módulo el sistema automáticamente le servirá tareas para realizar.

En la aplicación web existirá la posibilidad de los usuarios con perfil "Usuario", en caso de tener permisos, podrán conectarse a la cola de trabajo para que el sistema comience a servirles tareas a realizar.

3.4.11.1 Diagrama de casos de uso

En la siguiente figura aparece el diagrama de casos de uso que permite representar la conexión a colas de trabajo y el reparto automático de tareas:

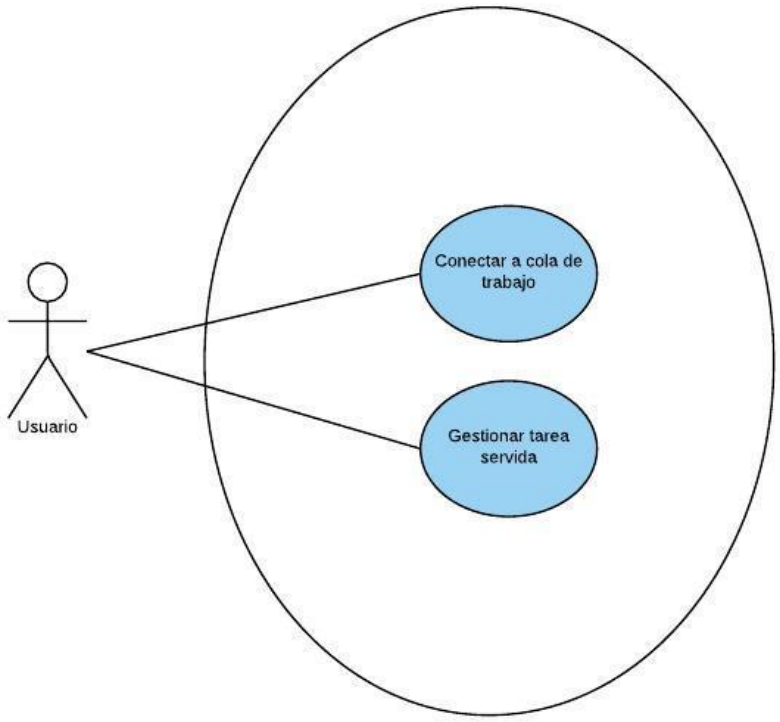

Figura 17. Diagrama de casos de uso "Reparto de Tareas"

3.4.11.2 Tareas asociadas a los requisitos PB-0-013

Una vez analizado los requisitos PB-0-013, se describirán las tareas a realizar en el desarrollo e implementación para el cumplimiento de estos requisitos:

- Capa de acceso a base de datos para la consulta de tareas dentro de las colas de trabajo.

- Servicio de la capa de negocio que se encargará de servir tareas de la cola de trabajo a la que se conecta el usuario siguiendo los criterios de filtrado y ordenación indicados en la cola de trabajo.

- Diseño de pantalla para la conexión de los usuarios a las colas de trabajo disponibles a las que tenga permisos.

3.4.12 Análisis del requisito PB-0-014

Según el contenido del Product Backlog elaborado en el punto 3.3.1, el requisito identificado como PB-0-014 indica lo siguiente:

La aplicación dispondrá de la posibilidad de asignación de tareas a un usuario en concreto. Cuando un usuario con permisos acceda a la gestión de una tarea podrá asignarla a cualquier otro usuario con permisos para que sea este último el único que pueda gestionarla.

En la aplicación web existirá la posibilidad de los usuarios con perfil "Usuario", en caso de tener permisos, podrán asignar tareas a otros usuarios para que sean estos últimos los únicos que puedan gestionar dicha tarea.

3.4.12.1 Diagrama de casos de uso

En la siguiente figura aparece el diagrama de casos de uso que permite representar la asignación de tareas:

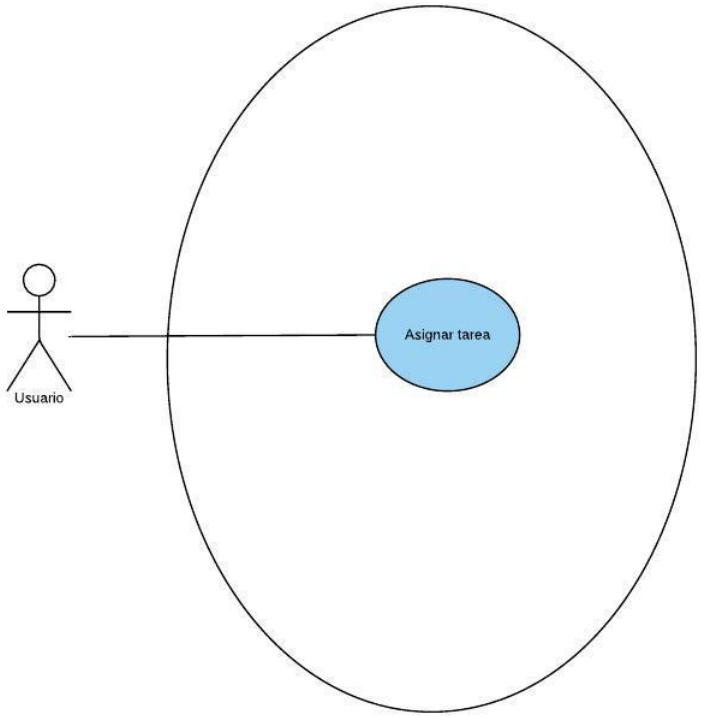

Figura 18. Diagrama de casos de uso "Asignación de Tareas"

3.4.12.2 Tareas asociadas a los requisitos PB-0-014

Una vez analizado los requisitos PB-0-014, se describirán las tareas a realizar en el desarrollo e implementación para el cumplimiento de estos requisitos:

- Capa de acceso a datos para la asignación de tareas a usuarios.
- Servicio de la capa de negocio que se encargará de gestionar las asignaciones de tareas a los distintos usuarios con permisos.
- Diseño de pantalla para la asignación de tareas a los usuarios.

3.5 Implementación de la aplicación web

3.5.1 Implementación del requisito PB-0-001

En el punto 3.4.2.1 habíamos descrito la arquitectura de la aplicación web, explicando el modelo de programación (modelo de programación por capas). Además habíamos descrito cada uno de los componentes que han sido necesarios para el desarrollo de la aplicación.

Para la implementación del requisito PB-0-001 vimos que era necesario la instalación y configuración de un servidor de aplicaciones en el que se desplegará la aplicación web y será accesible desde cualquier navegador web por los usuarios. En el punto 3.4.2.2 vimos la configuración del servidor Apache Tomcat 7.0.39 y la configuración del fichero de acceso a base de datos (MySQL 5.6)

Además de todo esto, en el punto 3.4.2.3, presentamos un modelo de pantalla que se utilizará como base para la definición del aspecto final de la aplicación. A partir de la plantilla diseñada se implementarán el conjunto de pantallas que compondrán la capa visual de la aplicación web. A continuación se muestra el aspecto final de la pantalla principal:

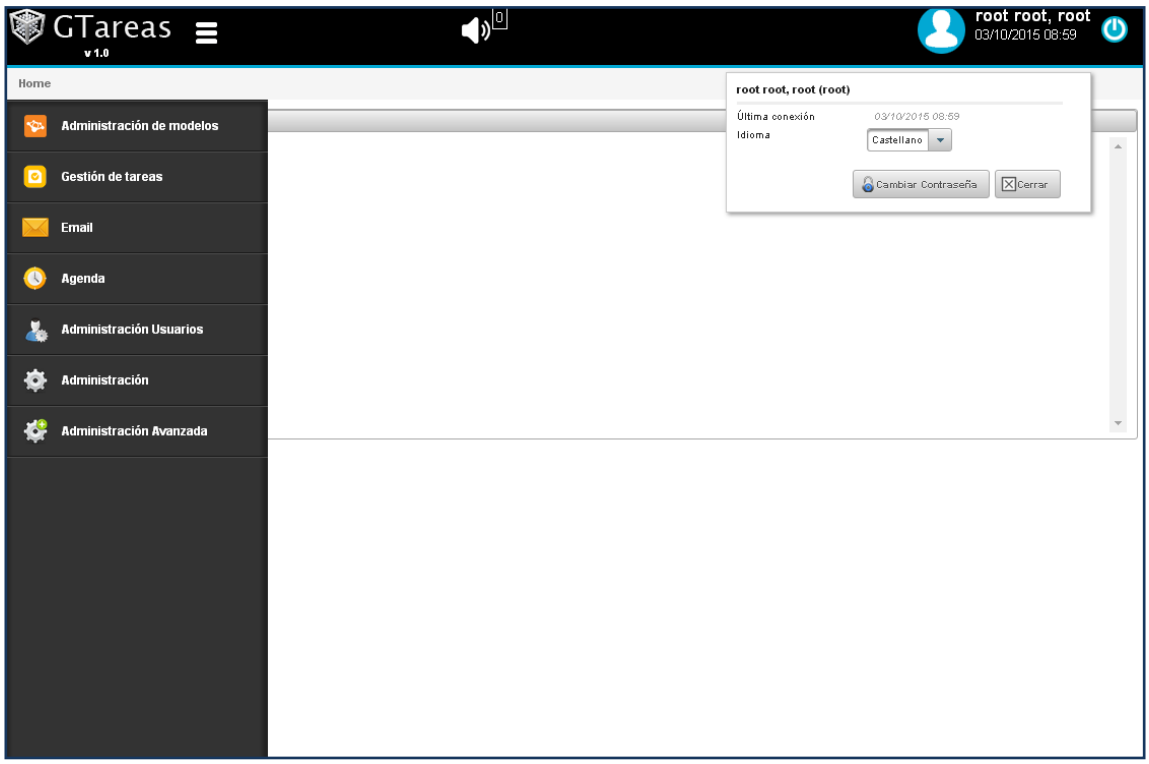

Figura 19. Aspecto final de la pantalla final de la aplicación "GTareas"

3.5.2 Implementación del requisito PB-0-002

En el punto 3.4.3 describimos las tareas necesarias para la implementación del requisito PB-0-002. Estas tareas son las siguientes:

- Diseño de esquema de base de datos con las tablas necesarias para el registro de usuarios
- Capa de acceso a base de datos tanto para el alta, la modificación y la baja de los usuarios así como la asignación de perfiles
- Servicio de la capa de negocio que se encargará tanto de la administración como de la autenticación de usuarios. Gestionará tanto la creación y mantenimiento de los usuarios como la comprobación de las credenciales de identificación de los usuarios que intenten acceder al sistema.
- Diseño de pantalla de administración de usuarios.
- Diseño de pantalla de login de acceso de usuarios.

3.5.2.1 Diseño de esquema de base de datos

En primer lugar definiremos las tablas necesarias para el registro de usuarios del sistema. En el diagrama entidad relación se indican las diferentes tablas definidas y las relaciones entre ellas. En cada tabla se indica su nombre, los campos y el tipo de datos de cada uno de ellos.

En el caso de las tablas necesarias para el registro de los usuarios se ha definido una sola tabla: *Usuario*.

- La tabla Usuario registrará los datos propios de la persona que accederá al sistema como nombre, apellidos, email, password, etc.

El diagrama entidad relación definido es el siguiente:

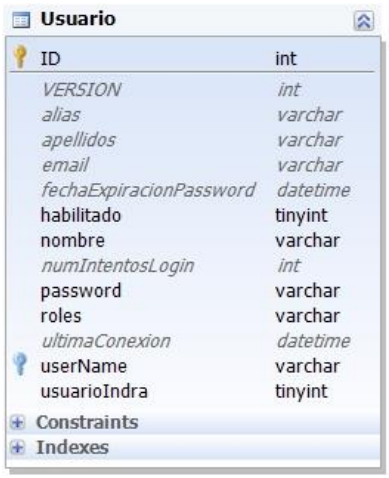

Figura 20. Diagrama entidad relación de tablas definidas para "Gestión de usuarios"

3.5.2.2 Acceso a datos para administración de usuario

A continuación describiremos la clase encargada del acceso a la base de datos, tanto para la consulta de datos como para la inserción, modificación o borrado de los registros. La clase llamada *DAOUsuario.java*[11] ofrece una serie de métodos para el alta, modificación y eliminación de usuarios. Los métodos más importantes de esta clase de la capa de acceso a datos son los siguientes:

createUsuario(Usuario usuario) Método que crea un objeto Usuario para la creación de un nuevo usuario
findAll() Método que recupera de la base de datos todos los usuarios.
findAllBy(UsuarioSpecification usuarioSpecification) Método que recupera de la base de datos los usuarios que cumplen los criterios de filtrado indicados en la clase *UsuarioSpecification* .
getByUsername(String nombre) Método que retorna de la base de datos el usuario de acuerdo al username dado.
getNumIntentosLogin(String userName) Método que retorna de la base de datos el número de intentos fallidos de acceso contabilizados al usuario con username dado
updateIncrementaNumIntentosLogin(String userName) Método que incrementa en una unidad el número de intentos fallidos de acceso del usuario con username dado actualizándolo en la base de datos
saveOrUpdate(Usuario usuario) Método que persiste en base de datos un Usuario. Si no existe en la base de datos lo inserta y si ya existe lo actualiza.

Tabla 26. Métodos más importantes de la clase DAOUsuario.java

3.5.2.3 Servicio de administración de usuario

A continuación describiremos la clase encargada de la lógica de negocio de la administración de usuarios. La clase llamada *ServicioGestionUsuario.java* ofrece una serie de métodos para el alta, modificación y eliminación de usuario. Los métodos más importantes del servicio de la capa de negocio son los siguientes:

createUsuario() Método que crea un nuevo usuario Indra con un Rol ya asignado.
getUsuarioBy(String username) Método que devuelve un usuario mediante un username dado

[11] Las clases de la capa de acceso a datos las denominaremos DAO, Data Access Object

isMaximosIntentosLoginSuperados(Usuario usuario) Método que indica si un usuario dado ha sobrepasado el número de intentos en el acceso al sistema
isUsuarioCreado(Usuario usuario) Método que se encarga de validar que un usuario no existe.
saveOrUpdate(Usuario usuario) Método que se encargar de hacer una llamada a la capa de acceso a datos para guardar un usuario en base de datos
txDelete(Usuario usuario) Método que se encarga de deshabilitar un usuario dado
txHabilitar(Usuario usuario) Método que se encarga de habilitar un usuario dado
updateDesbloquearUsuario(Usuario usuario) Método que se encarga de desbloquear un usuario dado que ha sido bloqueado por superar el número de intentos fallidos en el acceso al sistema
updateResetearUsuario(Usuario usuario, String password, boolean caducado) Método que se encarga de resetear la password de un usuario.

Tabla 27. Métodos más importantes de la clase ServicioGestionUsuario.java

3.5.2.4 Diseño de pantalla de administración de usuarios

Después de implementar las clases que gestionan el modelo de datos y la lógica de negocio, se procede a llevar a cabo la parte asociada a la vista (presentación) de la funcionalidad "Administración de usuarios".

Para implementar la capa de presentación se utiliza una clase java para la preparación y recogida de los datos de la vista y varios ficheros JSF donde se indica la composición de la vista que verá el usuario por pantalla.

La capa de presentación de la funcionalidad "Administración de usuarios" está compuesta por la clase *GestionUsuarioActionBean.java* y las vistas *listadoUsuarios.xhtml* y *editarUsuario.xhtml*.

La clase *GestionUsuarioActionBean.java* gestionará el objeto con los datos de la lista de usuarios y el objeto con el usuario seleccionado. La vista *listadoUsuarios.xhtml* mostrará por pantalla el listado de usuarios existentes en el sistema y la vista *editarUsuario.xhtml* mostrará los datos de un nuevo usuario o los datos del usuario seleccionado de la lista. Además ofrecerá las distintas opciones para gestionar al usuario: guardar datos, cambiar contraseña, etc.

A continuación se muestra el aspecto visual definitivo de las pantallas de administración de usuarios:

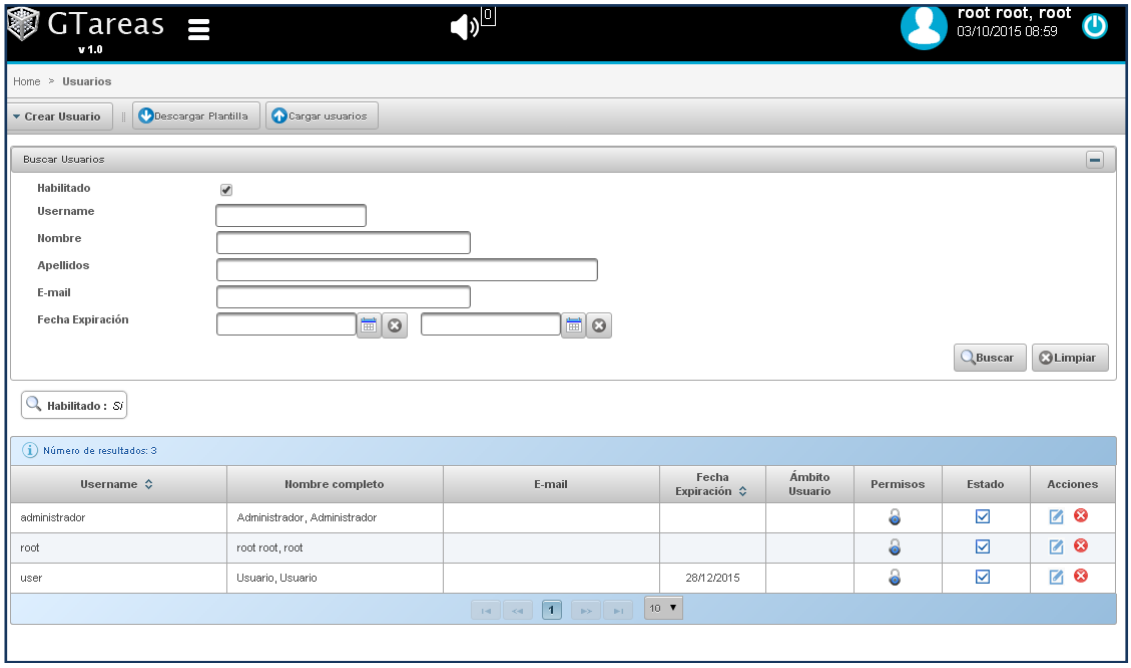

Figura 21. Aspecto final de la pantalla "listado de usuarios" de la aplicación "GTareas"

Al acceder a la opción de crear usuario o al editar un usuario existente se mostrará la pantalla de edición de usuario:

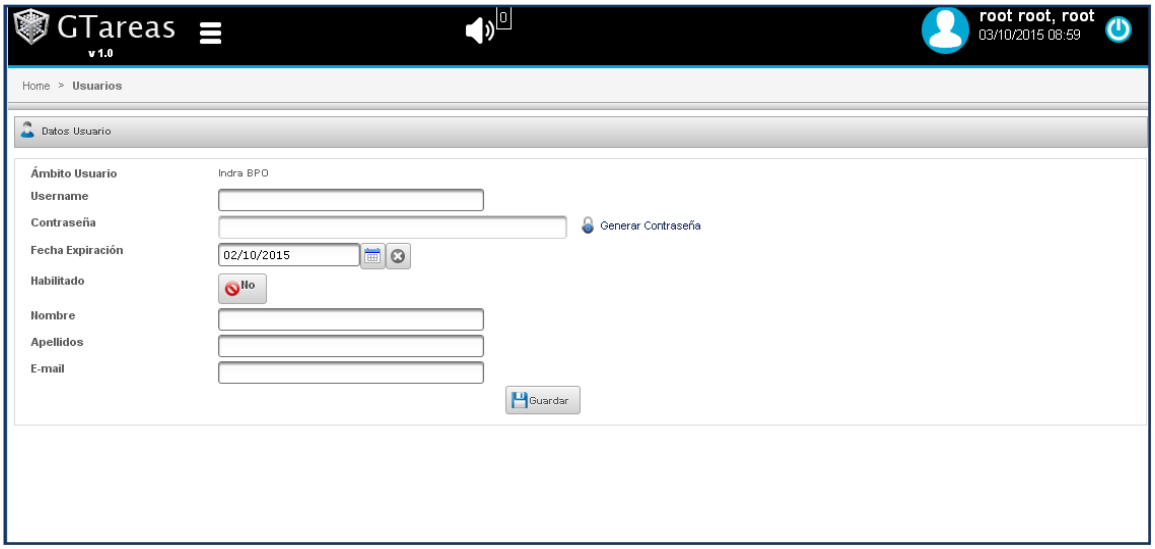

Figura 22. Aspecto final de la pantalla "edición de usuarios" de la aplicación "GTareas"

3.5.2.5 Diseño de pantalla de acceso a la aplicación

La clase *LoginActionBean.java* gestionará la presentación y la funcionalidad del acceso al sistema de los usuarios. Esta clase se ayudará de la parte de seguridad del framework *Spring*, denominado *Spring Security*. de La vista *home.xhtml* mostrará por pantalla de acceso al sistema donde se mostrará la posibilidad de indicar las credenciales de identificación (usuario y contraseña) al usuario.

A continuación se muestra el aspecto visual definitivo de la pantalla de acceso de usuarios:

Figura 23. Aspecto final de la pantalla "Acceso al sistema" de la aplicación "GTareas"

3.5.3 Implementación del requisito PB-0-003

En el punto 3.4.4 describimos las tareas necesarias para la implementación del requisito PB-0-003. Estas tareas son las siguientes:

- Diseño de esquema de base de datos con las tablas necesarias para el registro de perfiles.
- Capa de acceso a base de datos tanto para el alta, la modificación y la baja de los perfiles.
- Servicio de la capa de negocio que se encargará de la administración de los perfiles que estarán disponibles para su asignación a los usuarios
- Diseño de pantalla de administración de perfiles
- Diseño de pantalla de asignación de perfiles a los usuarios

3.5.3.1 Diseño de esquema de base de datos

En primer lugar definiremos las tablas necesarias para el registro de perfiles de usuario en el sistema. En la definición de las tablas se indicará el nombre de la tabla y los campos con su tipo de dato correspondiente.

En el caso de las tablas necesarias para el registro de los los perfiles se han definido 2 tablas: *UsuarioPerfil y UsuarioUsuarioPerfil*.

- La tabla *UsuarioPerfil* registrará los distintos perfiles existentes en el sistema. Se almacenará el nombre del perfil y los roles que contendrá este perfil.

- La tabla *UsuarioUsuarioPerfil* registrará los perfiles que tiene cada usuario en el sistema.

El diagrama entidad relación definido es el siguiente:

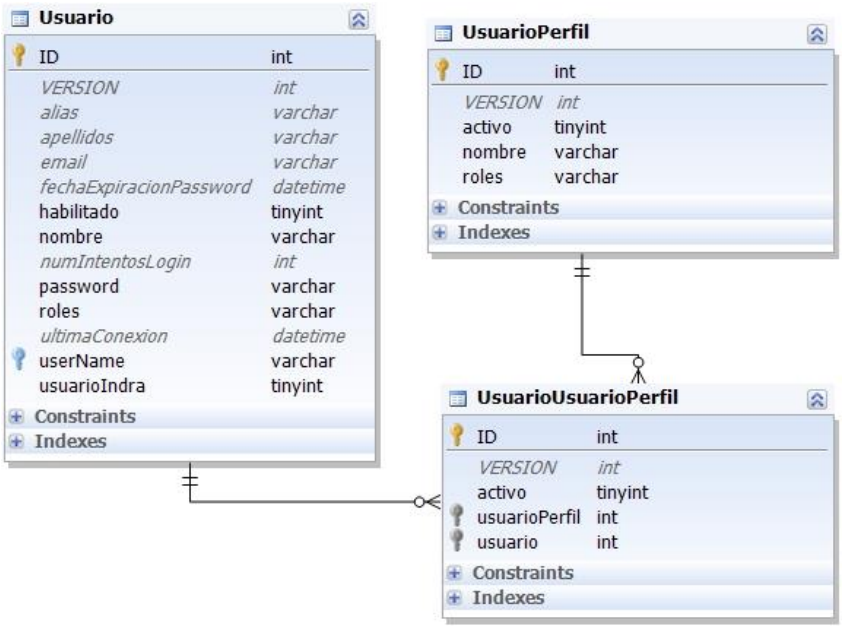

Figura 24. Diagrama entidad relación de tablas definidas para "Gestión de perfiles"

3.5.3.2 Acceso a datos para administración de perfiles

A continuación describiremos la clase encargada del acceso a la base de datos, tanto para la consulta de datos como para la inserción, modificación o borrado de los perfiles de usuario.

La clase *DAOUsuarioPerfil.java* ofrece una serie de métodos para el alta, modificación y eliminación de perfiles. La clase *DAOUsuarioUsuarioPerfil.java* nos ofrece una serie de métodos necesarios para el registro en base de datos de los perfiles asignados a los distintos usuarios.

Los métodos más importantes de la clase *DAOUsuarioPerfil.java* (capa de acceso a datos) son los siguientes:

createUsuarioPerfil()
Método que crea un objeto UsuarioPerfil con los datos del perfil a crear en la base de datos.
findAll()

Método que recupera de la base de datos todos los perfiles.
findAllBy(UsuarioPerfilSpecification usuarioPerfilSpecification)
Método que recupera de la base de datos los perfiles que cumplen los criterios de filtrado indicados en la clase *UsuarioPerfilSpecification* .
saveOrUpdate(UsuarioPerfil usuarioPerfil)
Método que inserta o actualiza un perfil en la base de datos.

Tabla 28. Métodos más importantes de la clase DAOUsuarioPerfil.java

Los métodos más importantes de la clase *DAOUsuarioUsuarioPerfil.java* (capa de acceso a datos) son los siguientes:

createUsuarioUsuarioPerfil ()
Método que crea un objeto UsuarioUsuarioPerfil con los datos del usuario y el perfil.
deleteUsuarioUsuarioPerfil(Integer usuarioId, Integer usuarioPerfilId)
Método que elimina de la base de datos el perfil con identificador *usuarioPerfilId* del usuario con identificador *usuarioId*.
findUsuarioPerfilBy(Usuario usuario)
Método que recupera de la base de datos los perfiles de un usuario dado.
saveOrUpdate(UsuarioUsuarioPerfil usuarioUsuarioPerfil)
Método que inserta o actualiza en la base de datos el perfil de un usuario.

Tabla 29. Métodos más importantes de la clase DAOUsuarioUsuarioPerfil.java

3.5.3.3 Servicio de administración y asignación de perfiles

A continuación describiremos la clase encargada de la lógica de negocio de la administración y asignación de perfiles a usuarios. La clase llamada *ServicioGestionUsuarioPerfil.java* ofrece una serie de métodos para el alta, modificación y eliminación de perfiles en el sistema, así como métodos para la asignación o eliminación de perfiles a los usuarios. Los métodos más importantes del servicio de la capa de negocio *ServicioGestionUsuarioPerfil.java* son los siguientes:

createUsuarioPerfil()
Metodo que crea un objeto UsuarioPerfil con los datos del perfil
createUsuarioUsuarioPerfil ()
Método que crea un objeto UsuarioUsuarioPerfil con los datos del usuario y el perfil.
deletePerfil(UsuarioUsuarioPerfil usuarioUsuarioPerfil)
Método que elimina un perfil de un usuario mediante el objeto UsuarioUsuarioPerfil.
deletePerfil(Integer usuarioId, Integer usuarioPerfilId)

Método que elimina un perfil de un usuario mediante el identificador del perfil y del usuario..
findUsuarioPerfilBy(Usuario usuario) Método que devuelve la lista de perfiles que dispone un usuario dado.
saveUsuarioPerfil(UsuarioPerfil usuarioPerfil) Método que hace la llamada a la capa de acceso a datos para guardar un perfil en el sistema.
savePerfil(UsuarioUsuarioPerfil usuarioUsuarioPerfil) Metodo que hace la llamada a la capa de acceso a datos para asignar un perfil a un usuario

Tabla 30. Métodos más importantes de la clase ServicioUsuarioPerfil.java

3.5.3.4 Diseño de pantalla de administración y asignación de perfiles

La capa de presentación de la funcionalidad "Administración de perfiles de usuario" está compuesta por la clase *GestionUsuarioPerfilActionBean.java* y las vistas *listadoUsuarioPerfil.xhtml*, *editarPerfil.xhtml* y *includeAsignarPerfil.xhtml*.

La clase *GestionUsuarioPerfilActionBean.java* gestionará el objeto con los datos de la lista de perfiles y el objeto con el perfil seleccionado. La vista *listadoUsuarioPerfil.xhtml* mostrará por pantalla el listado de perfiles existentes en el sistema y la vista *editarPerfil.xhtml* mostrará los datos de un nuevo perfil o los datos del perfil seleccionado de la lista. Además ofrecerá las distintas opciones para gestionar un perfil: guardar datos, asignar roles (permisos) al perfil, etc.

A continuación se muestra el aspecto visual definitivo de las pantallas de administración de perfiles:

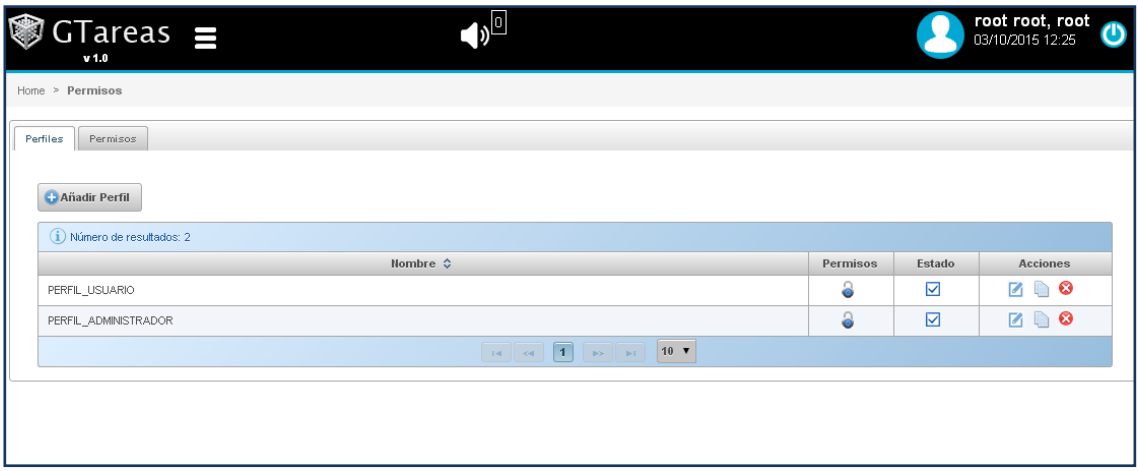

Figura 25. Aspecto final de la pantalla "listado de perfiles" de la aplicación "GTareas"

Al acceder a la opción de añadir perfil o al editar un perfil existente se mostrará la pantalla de edición de perfil, *editarPerfil.xhtml*, donde se podrá indicar/modificar el

nombre y agregar o eliminar los roles (permisos) asignados al perfil. El aspecto visual de pantalla definitiva de la edición de un perfil es la siguiente:

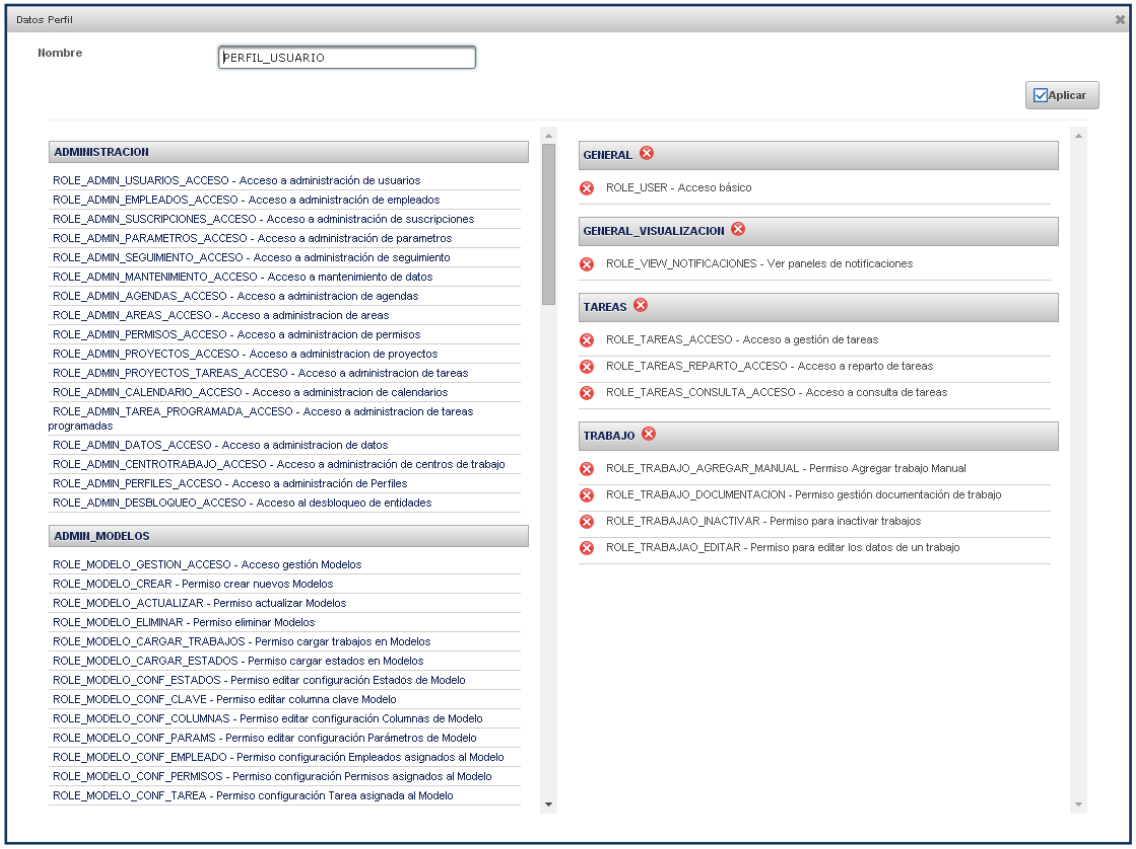

Figura 26. Aspecto final de la pantalla "edición de perfil" de la aplicación "GTareas"

Una vez definidos los perfiles necesarios dentro del sistema, necesitamos un módulo de asignación de perfiles a un usuario determinado. La porción de pantalla donde se puede asignar un perfil a un usuario es *includeAsignarPerfil.xhtml*. Esta porción de pantalla la incluiremos en la edición de un usuario para así integrar la gestión de los datos del usuario con la lista de perfiles asignados. A continuación se muestra la vista definitiva de la gestión de un usuario con la lista de perfiles y la posibilidad de agregar o eliminar perfiles de dicha lista:

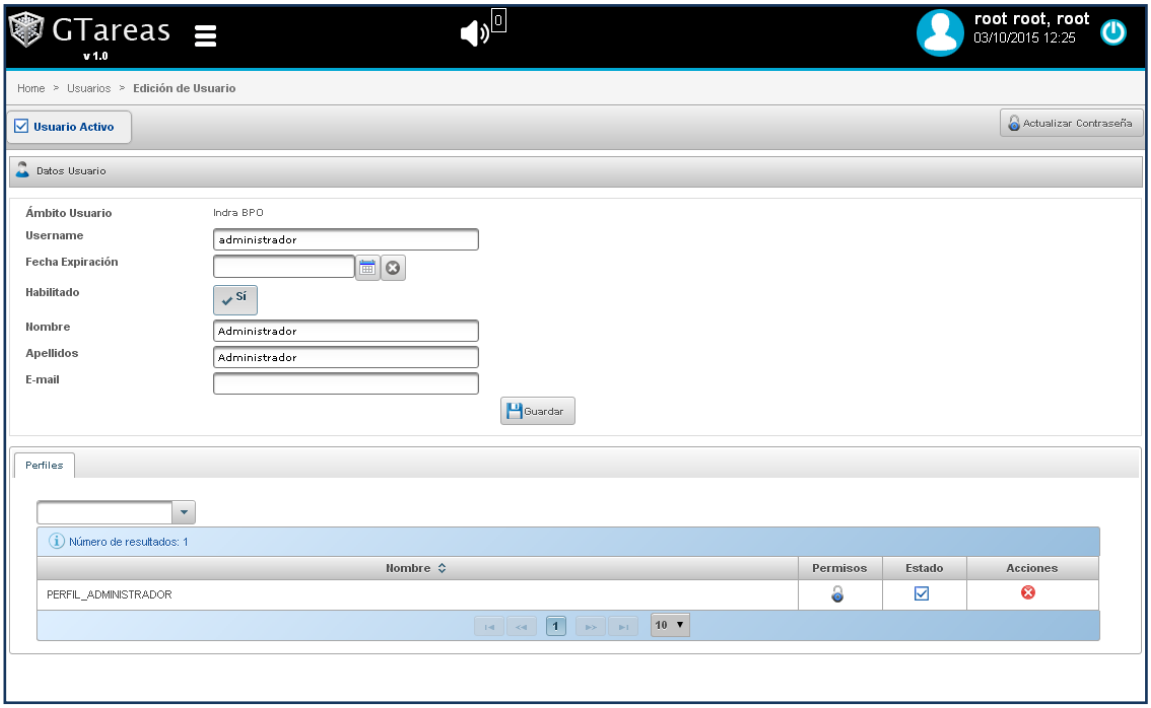

Figura 27. Aspecto final de la pantalla "edición de usuario" con la lista de perfiles

3.5.4 Implementación del requisito PB-0-004 al PB-0-007

En el punto 3.4.5 describimos las tareas necesarias para la implementación de los requisitos PB-0-004, PB-0-005, PB-0-006 y PB-0-007. Estas tareas son las siguientes:

- Diseño de esquema de base de datos con las tablas necesarias para el registro de modelos de trabajo, además del registro de campos, estados y permisos para estos modelos de trabajo.

- Capa de acceso a base de datos tanto para el alta, la modificación y la baja de los modelos de trabajo, así como para el registro de los estados, campos y permisos.

- Servicio de la capa de negocio que se encargará de la administración de los modelos de trabajo, sus estados, sus campos y sus permisos.

- Diseño de pantalla de administración de modelos de trabajo.

- Diseño de pantalla para el mantenimiento de estados por los que podrán pasar las tareas que pertenezcan al modelo de trabajo.

- Diseño de pantalla para el mantenimiento de campos donde se guardará la información necesaria de las tareas que pertenezcan al modelo de trabajo.

- Diseño de pantalla para el mantenimiento de los usuarios con permisos para gestionar las tareas que pertenezcan al modelo de trabajo.

3.5.4.1 Diseño de esquema de base de datos

En primer lugar definiremos las tablas necesarias para el registro de modelos de trabajo, además del registro de campos, estados y permisos para estos modelos de trabajo. En la definición de las tablas se indicará el nombre de la tabla y los campos con su tipo de dato correspondiente.

En este caso se han definido 4 tablas: *TrabajoModelo*, *TrabajoModeloEstado*, *TrabajoModeloEstadoRelacion* y *TrabajoModeloEmpleado*.

- La tabla *TrabajoModelo* registrará los datos del modelo de trabajo (nombre y los campos que contiene).

- La tabla *TrabajoModeloEstado* registrará los datos de los distintos estados del modelo de trabajo. La tabla *TrabajoModeloEstadoRelacion* registrará la relación existente entre los estados y si son estados finales o no.

- La tabla *TrabajoModeloEmpleado* registrará los usuarios que tienen permisos dentro del modelo de trabajo

El diagrama entidad relación definido es el siguiente:

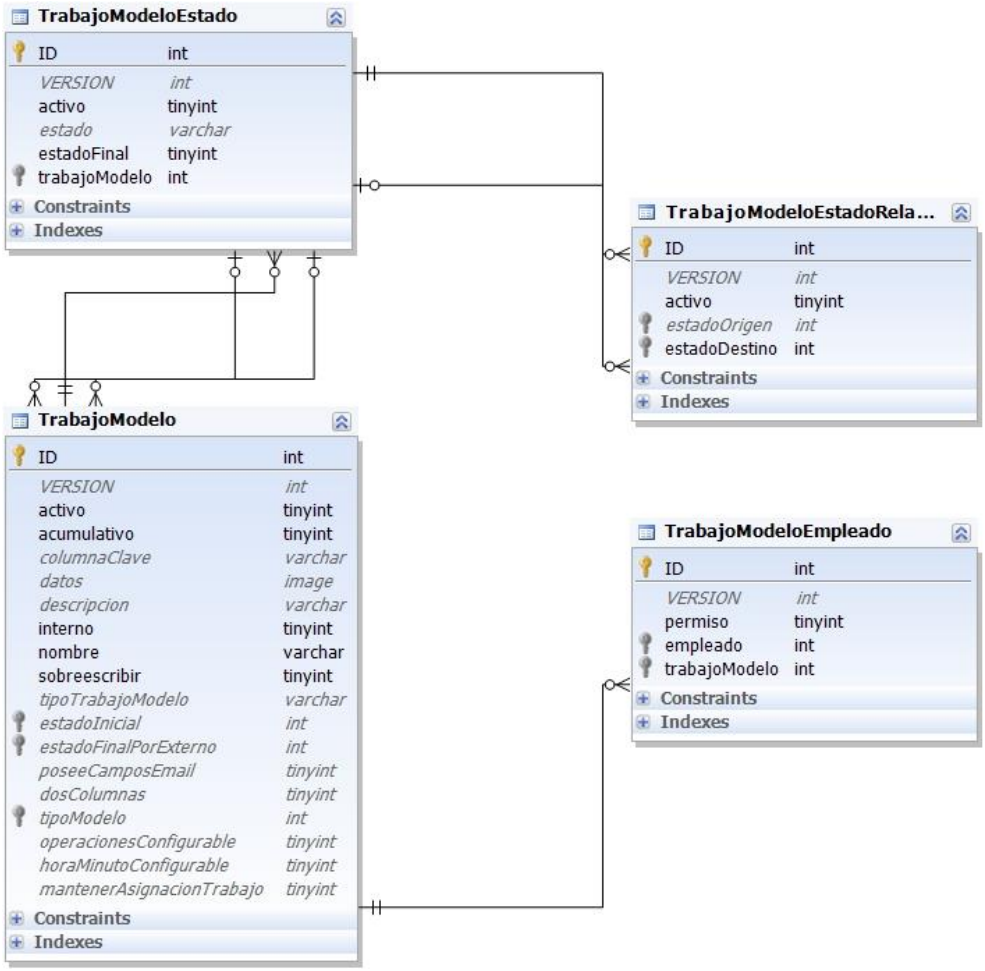

Figura 28. Diagrama entidad relación de tablas definidas para "Gestión de modelos de trabajo"

3.5.4.2 Acceso a datos para administración de modelos de trabajo

A continuación describiremos las clases encargadas del acceso a la base de datos, tanto para la consulta de datos como para la inserción, modificación o borrado de los modelos de trabajo, sus estados y relaciones y los permisos necesarios:

- La clase *DAOTrabajoModelo.java* ofrece una serie de métodos la inserción, modificación y eliminación en base de datos los de modelos de trabajo. Además ofrece los métodos necesarios para la administración de los campos de un modelo de trabajo.

- La clase *DAOTrabajoModeloEstado.java* nos ofrece una serie de métodos necesarios para el registro en base de datos de los estados asignados a cada modelo de trabajo. La clase *DAOTrabajoModeloEstadoRelacion.java* nos ofrece una serie de métodos necesarios para el registro en base de datos de las relaciones existentes entre los estados asignados a un mismo modelo de trabajo.

- La clase *DAOTrabajoModeloEmpleado.java* nos ofrece una serie de métodos necesarios para el registro en base de datos de los permisos de los usuarios dentro de cada modelo de trabajo.

Los métodos más importantes de la clase *DAOTrabajoModelo.java* (capa de acceso a datos) son los siguientes:

createTrabajoModelo()
Método que crea un objeto TrabajoModelo con los datos del modelo de trabajo a crear en la base de datos.
findAll()
Método que recupera de la base de datos todos los modelos de trabajo.
findAllActivos()
Método que recupera de la base de datos todos los modelos de trabajo que están activos.
saveOrUpdate(TrabajoModelo trabajoModelo)
Método que inserta o actualiza un perfil en la base de datos.
desActivar(TrabajoModelo trabajoModelo)
Método que actualiza un modelo de trabajo en la base de datos con la propiedad “activo” tomando el valor “false”.

Tabla 31. Métodos más importantes de la clase DAOTrabajoModelo.java

Los métodos más importantes de la clase *DAOTrabajoModeloEstado.java* (capa de acceso a datos) son los siguientes:

createTrabajoModeloEstado()
Método que devuelve un objeto con los datos del estado y el modelo trabajo al que pertenece.
findEstadosByModelo(TrabajoModelo trabajoModelo)

Método que recupera de la base de datos la lista de estados de un modelo de trabajo dado.
findEstadosByModeloEstadoNombre(TrabajoModelo trabajoModelo, String estadoNombre)
Método que recupera de la base de datos la lista de estados de un modelo de trabajo dado y el nombre del estado.
saveOrUpdate(TrabajoModeloEstado trabajoModeloEstado)
Método que inserta o actualiza en la base de datos el estado de un modelo de trabajo.
deleteByTrabajoModelo(TrabajoModelo trabajoModelo)
Método que elimina de la base de datos todos los estados de un modelo de trabajo dado.
delete(TrabajoModeloEstado trabajoModeloEstado)
Método que elimina de la base de datos un estado dado.

Tabla 32. Métodos más importantes de la clase DAOTrabajoModeloEstado.java

Los métodos más importantes de la clase *DAOTrabajoModeloEstadoRelacion.java* (capa de acceso a datos) son los siguientes:

createTrabajoModeloEstadoRelacion()
Método que devuelve un nuevo objeto con los datos de la relación entre estados.
findRelacionesEstadoByEstadoOrigen(TrabajoModeloEstado trabajoModeloEstado)
Método que recupera de la base de datos la lista de relaciones entre estados de un estado dado.
findRelacionesEstadoByModelo(TrabajoModelo trabajoModelo)
Método que recupera de la base de datos la lista de las relaciones entre estados de un modelo de trabajo dado.
saveOrUpdate(TrabajoModeloEstadoRelacion trabajoModeloEstadoRelacion)
Método que inserta o actualiza en la base de datos una relación entre estados dada.
delete(TrabajoModeloEstadoRelacion trabajoModeloEstadoRelacion)
Método que elimina de la base de datos una relación entre estados dada.

Tabla 33. Métodos más importantes de la clase DAOTrabajoModeloEstadoRelacion.java

Los métodos más importantes de la clase *DAOTrabajoModeloEmpleado.java* (capa de acceso a datos) son los siguientes:

createTrabajoModeloEmpleado()
Método que devuelve un nuevo objeto con los datos del usuario y el modelo trabajo al que se le otorgan permisos de acceso y gestión.
findEmpleadoBy (TrabajoModelo trabajoModelo)
Método que recupera de la base de datos la lista de usuarios con permisos dentro de un modelo de trabajo dado.
saveOrUpdate (TrabajoModeloEmpleado trabajoModeloEmpleado)
Método que inserta o actualiza en la base de datos un usuario dentro de un modelo de trabajo.

updatePermisoAdministrador (TrabajoModelo trabajoModelo, Usuario usuario)
Método que actualiza en la base de datos los permisos de administrador de un usuario dentro de un modelo de trabajo.
deleteAllEmpleadoBy(Integer modeloTrabajoId)
Método que elimina de la base de datos todos los usuarios de un modelo de trabajo dado por el identificador.
deleteTrabajoModeloEmpleado(Integer empleadoId, Integer modeloTrabajoId)
Método que elimina de la base de datos un usuario de un modelo de trabajo ambos dados por el identificador.
isEmpleadoAdmin(Integer trabajoModeloId, Integer empleadoId)
Método que tras consultar en la base de datos indica si un usuario dado es administrador de un modelo de trabajo
isEmpleadoUsuario (Integer trabajoModeloId, Integer empleadoId)
Método que tras consultar en la base de datos indica si un usuario dado es usuario de un modelo de trabajo.

Tabla 34. Métodos más importantes de la clase DAOTrabajoModeloEmpleado.java

3.5.4.3 Servicio de administración de modelos de trabajo

Los servicios donde se define la lógica de negocio para la administración de los modelos de trabajo son:

- La clase *ServicioTrabajoModelo.java* ofrece una serie de métodos para el alta, modificación y eliminación de modelos de trabajo. Además ofrece los métodos necesarios para la administración de los campos de un modelo de trabajo.

- La clase *ServicioTrabajoModeloEstado.java* nos ofrece una serie de métodos necesarios para la gestión de estados dentro de un modelo de trabajo. La clase *ServicioTrabajoModeloEstadoRelacion.java* nos ofrece una serie de métodos necesarios para la gestión de las relaciones existentes entre los estados asignados a un mismo modelo de trabajo.

- La clase *ServicioTrabajoModeloEmpleado.java* nos ofrece una serie de métodos necesarios para la gestión de los permisos de los usuarios dentro de cada modelo de trabajo.

Los métodos más importantes de la clase *ServicioTrabajoModelo.java* (capa de lógica de negocio de la aplicación) son los siguientes:

createTrabajoModelo()
Método que crea un nuevo objeto TrabajoModelo con los datos del modelo de trabajo a crear.
findAll()
Método que obtiene todos los modelos de trabajo.
findAllActivos()

Método que obtiene todos los modelos de trabajo que están activos.
saveOrUpdate(TrabajoModelo trabajoModelo)
Método que envía a la capa de acceso a datos el objeto *TrabajoModelo* preparado para ser registrado.
txEliminar(TrabajoModelo trabajoModelo)
Método que envía a la capa de acceso a datos el objeto TrabajoModelo que ha de ser eliminado.
txCargarCamposModeloTrabajo (TrabajoModelo trabajoModelo, File excel)
Método que crea la estructura de campos de un modelo de trabajo desde un fichero excel.
txAddCampoModeloTrabajo (TrabajoModelo trabajoModelo, CampoModelo campoModelo)
Método que crea un nuevo campo dentro de un modelo de trabajo.

Tabla 35. Métodos más importantes de la clase ServicioTrabajoModelo.java

Los métodos más importantes de la clase *ServicioTrabajoModeloEstado.java* (capa de lógica de negocio de la aplicación) son los siguientes:

createTrabajoModeloEstado()
Método que crea un nuevo objeto TrabajoModeloEstado con los datos del estado y el modelo trabajo al que pertenece.
findEstadosByModelo(TrabajoModelo trabajoModelo)
Método que obtiene la lista de estados de un modelo de trabajo dado.
findEstadosByModeloEstadoNombre(TrabajoModelo trabajoModelo, String estadoNombre)
Método que obtiene la lista de estados de un modelo de trabajo dado y el nombre del estado.
saveOrUpdate(TrabajoModeloEstado trabajoModeloEstado)
Método que registra el estado de un modelo de trabajo.
deleteByTrabajoModelo(TrabajoModelo trabajoModelo)
Método que elimina todos los estados de un modelo de trabajo dado.
txEliminar(TrabajoModeloEstado trabajoModeloEstado)
Método que elimina un estado dado.

Tabla 36. Métodos más importantes de la clase ServicioTrabajoModeloEstado.java

Los métodos más importantes de la clase *ServicioTrabajoModeloEstadoRelacion.java* (capa de lógica de negocio de la aplicación) son los siguientes:

createTrabajoModeloEstadoRelacion()
Método que crea un nuevo objeto con los datos de la relación entre estados.
findRelacionesEstadoByEstadoOrigen(TrabajoModeloEstado trabajoModeloEstado)
Método que devuelve la lista de relaciones entre estados de un estado dado.
findRelacionesEstadoByModelo(TrabajoModelo trabajoModelo)

Método que devuelve la lista de las relaciones entre estados de un modelo de trabajo dado.
saveOrUpdate(TrabajoModeloEstadoRelacion trabajoModeloEstadoRelacion)
Método que registra una relación entre estados dada.
delete(TrabajoModeloEstadoRelacion trabajoModeloEstadoRelacion)
Método que elimina una relación entre estados dada.

Tabla 37. Métodos más importantes de la clase ServicioTrabajoModeloEstadoRelacion.java

Los métodos más importantes de la clase *ServicioTrabajoModeloEmpleado.java* (capa de lógica de negocio de la aplicación) son los siguientes:

createTrabajoModeloEmpleado()
Método que crea un nuevo objeto con los datos del usuario y el modelo trabajo al que se le otorgan permisos de acceso y gestión.
findEmpleadoBy (TrabajoModelo trabajoModelo)
Método que devuelve la lista de usuarios con permisos dentro de un modelo de trabajo dado.
saveOrUpdate (TrabajoModeloEmpleado trabajoModeloEmpleado)
Método que registra un usuario dentro de un modelo de trabajo.
updatePermisoAdministrador (TrabajoModelo trabajoModelo, Usuario usuario)
Método que registra los permisos de administrador de un usuario dentro de un modelo de trabajo.
deleteAllEmpleadoBy(Integer modeloTrabajoId)
Método que elimina todos los usuarios de un modelo de trabajo dado por el identificador.
deleteTrabajoModeloEmpleado(Integer empleadoId, Integer modeloTrabajoId)
Método que elimina un usuario de un modelo de trabajo ambos dados por el identificador.
isEmpleadoAdmin(Integer trabajoModeloId, Integer empleadoId)
Método que indica si un usuario dado es administrador de un modelo de trabajo
isEmpleadoUsuario (Integer trabajoModeloId, Integer empleadoId)
Método que indica si un usuario dado es usuario de un modelo de trabajo.

Tabla 38. Métodos más importantes de la clase ServicioTrabajoModeloEmpleado.java

3.5.4.4 Diseño de pantalla de administración de modelos de trabajo

La capa de presentación de la funcionalidad "Administración de modelos de trabajo" está compuesta por la clase *GestionModeloTrabajoActionBean.java* y las vistas *gestionModeloTrabajo.xhtml*.

La clase *GestionModeloTrabajoActionBean.java* gestionará el objeto con los datos de la lista de modelos de trabajo y el objeto con el modelo de trabajo seleccionado. La vista *gestionModeloTrabajo.xhtml* mostrará por pantalla un modelo de trabajo nuevo o el modelo de trabajo seleccionado después de que el usuario escoja una opción u otra. Dentro de la pantalla de gestión del modelo de trabajo, aparte de los datos propios del

modelo de trabajo, se mostrarán varias pestañas con los datos de la configuración de estados y sus relaciones, los campos del modelo y los permisos de los usuarios dentro del modelo, todo ello con los mecanismos necesarios para su gestión.

A continuación se muestra el aspecto visual definitivo de las pantallas de administración de modelos de trabajo:

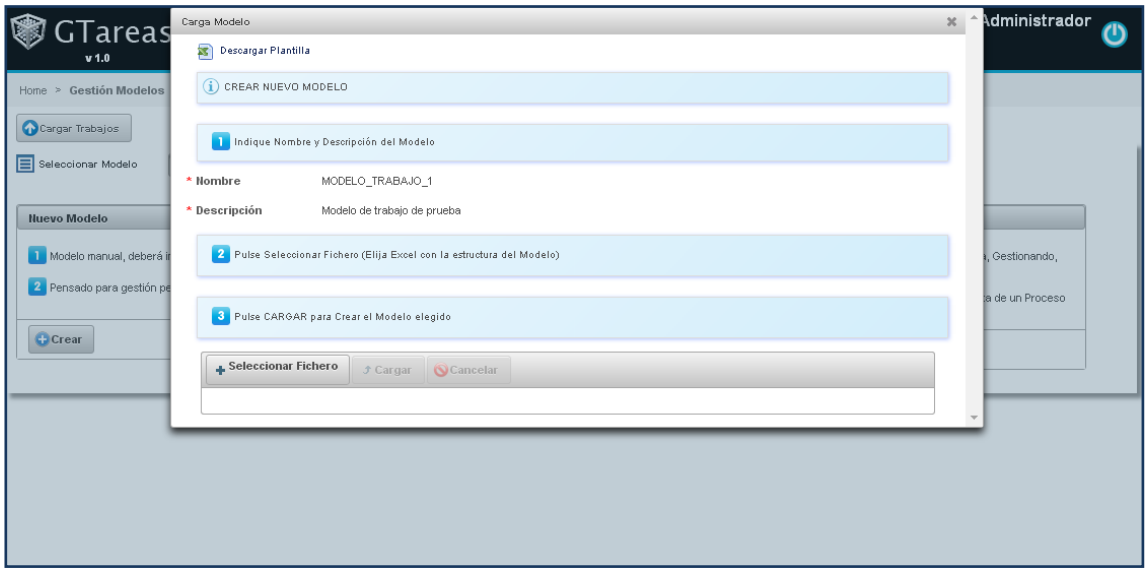

Figura 29. Aspecto visual de la pantalla "crear modelo de trabajo"

Observamos en la figura anterior la pantalla para crear un nuevo modelo de trabajo. Es necesario indicar el nombre que identificará al modelo, una descripción y cargar un fichero Excel donde se indicarán todos los campos para almacenar los datos de cada tarea perteneciente al modelo.

Una vez realizada el alta de un nuevo modelo de trabajo siguiendo los pasos descritos en el apartado anterior, aparecerá automáticamente la pantalla de administración del modelo de trabajo creado:

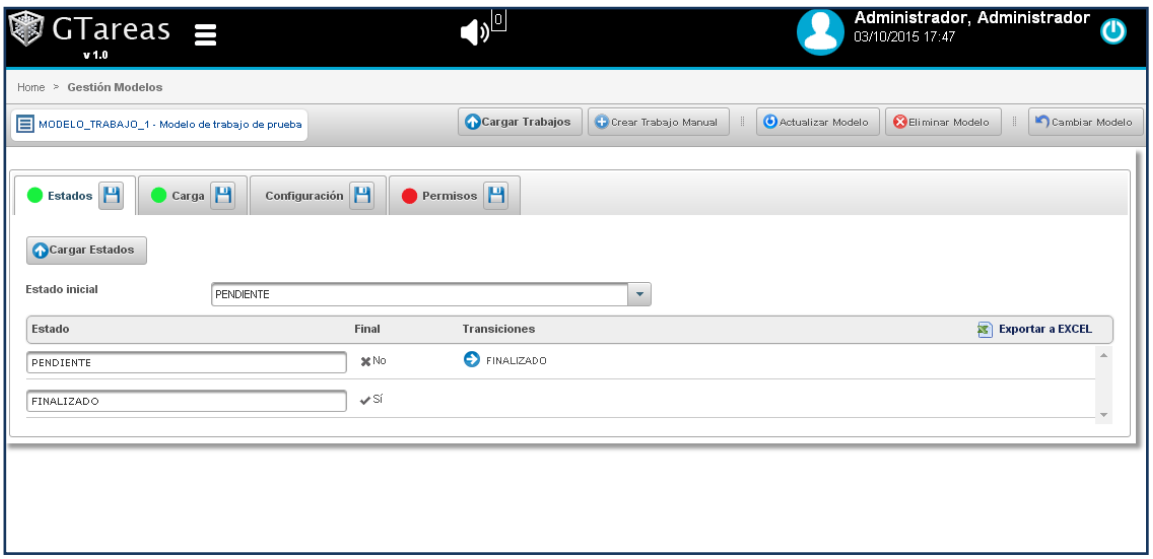

Figura 30. Aspecto visual de la pantalla "administración de estados del modelo de trabajo"

Observamos las distintas pestañas de configuración del modelo de trabajo. La primera pestaña (la que observamos en la figura anterior) se encarga de la configuración de los estados y sus relaciones. Desde esta parte de la administración del modelo se configuran los distintos estados del modelo y la transición entre ellos. Se podrá cargar esta información desde un fichero Excel en el que se indicará una fila por estado y en las columnas las distintas transiciones entre estados.

La segunda pestaña está destinada a la carga de tareas de este modelo de trabajo. La tercera pestaña a la configuración de los campos del modelo de trabajo y la cuarta pestaña para la configuración de permisos de acceso y gestión del modelo de trabajo.

A continuación se muestra la pestaña de configuración de los campos del modelo de trabajo:

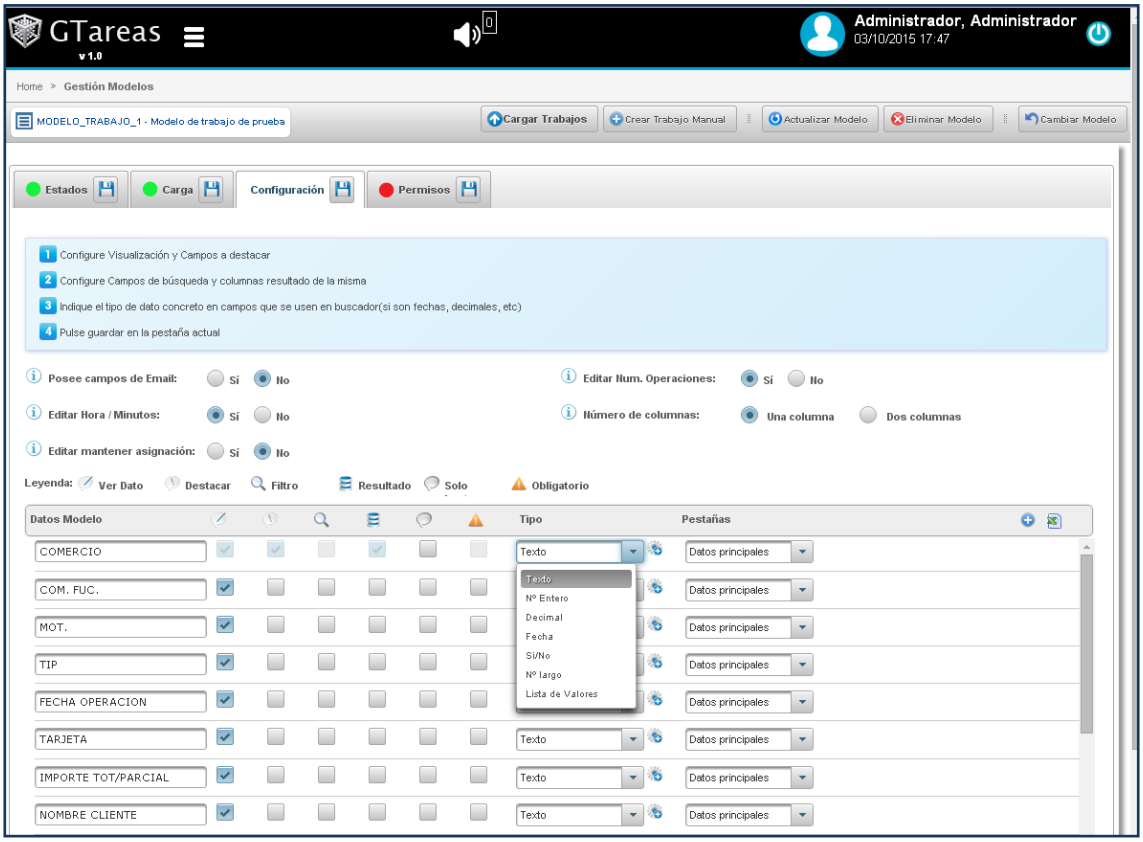

Figura 31. Aspecto visual de la pantalla "administración de campos del modelo de trabajo"

Observamos las distintas opciones de configuración de los campos del modelo de trabajo junto con las opciones de visualización dentro de la tarea. Por cada campo se indica si un campo se visualizará en la ficha de la tarea, si se deberá mostrar de forma destacada, si se deberá visualizar en el filtro de la búsqueda de las tareas, si deberá aparecer en el resultado de la búsqueda o si al campo deberá asignársele un valor obligatoriamente. Además se deberá seleccionar el tipo de dato de cada campo, indicando si el campo es de tipo texto, numero entero, numero decimal, numero largo, fecha, tipo Si/No o un valor dentro de una lista de valores.

A continuación se muestra la pestaña de permisos del modelo de trabajo, donde se indicarán los usuarios que tendrán permisos para administrar el modelo así como los usuarios con permisos para actuar sobre las tareas del modelo:

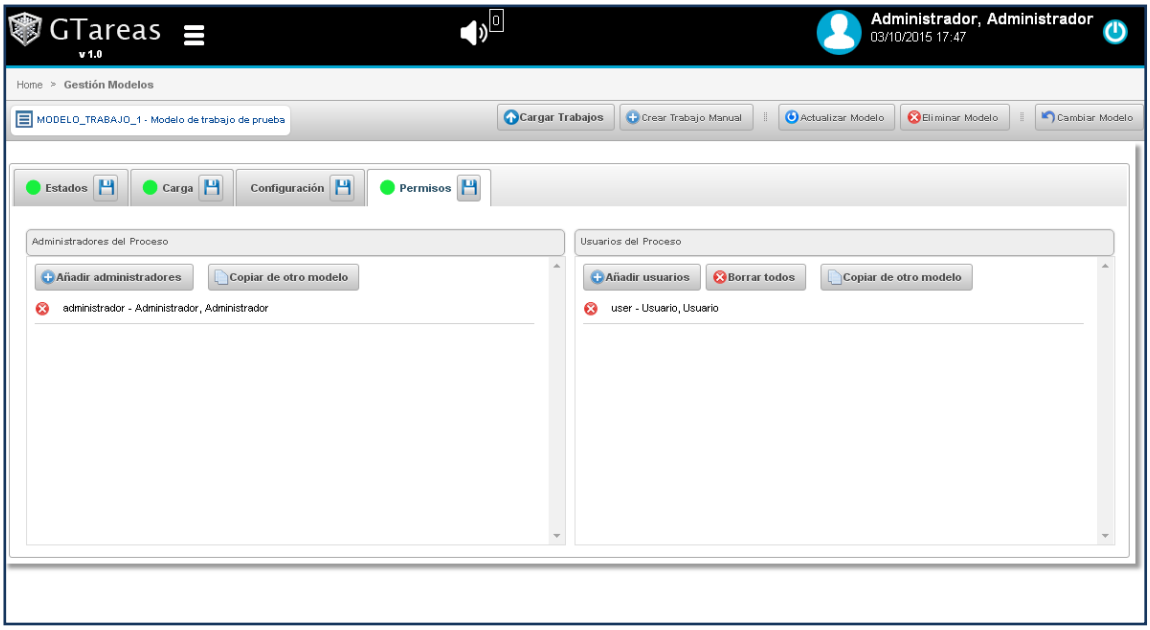

Figura 32. Aspecto visual de la pantalla "administración de permisos del modelo de trabajo"

Con este paso ya habremos concluido de configurar el modelo de trabajo. Ahora el siguiente paso sería cargar tareas de este modelo o bien manualmente o bien mediante la carga de un fichero.

3.5.5 Implementación del requisito PB-0-008

En el punto 3.4.6 describimos las tareas necesarias para la implementación del requisito PB-0-008. Estas tareas son las siguientes:

- Diseño de esquema de base de datos con las tablas necesarias para el registro de tareas de un modelo de trabajo.
- Servicio de la capa de negocio que se encargará de la lectura del fichero Excel y la adaptación de los datos recibidos.
- Capa de acceso a base de datos para el registro de las tareas.
- Diseño de pantalla para la carga de tareas desde un fichero Excel.

3.5.5.1 Diseño de esquema de base de datos

En primer lugar definiremos las tablas necesarias para el registro de tareas de un modelo de trabajo. En la definición de las tablas se indicará el nombre de la tabla y los campos con su tipo de dato correspondiente.

En este caso se han definido las tablas: *Trabajo*, *TrabajoSeguimiento*, *TrabajoModeloDato* y *Bloqueo:*

- La tabla *Trabajo* registrará los datos de la tarea que pertenecerá a un modelo de trabajo.
- La tabla *TrabajoSeguimiento* registrará todas las acciones realizadas sobre la tarea. Se almacenará la fecha y hora de la acción, el usuario que realiza la acción, el tiempo que ha tardado y las características de la acción (consulta, cambio de estado, etc.)
- La tabla *TrabajoModeloDato* registrará los valores de los campos de la tarea.
- La tabla *Bloqueo* registrará todas aquellas tareas que han sido bloqueadas, tanto si están trabajando con ellas como si han sido asignadas a algún usuario.

El diagrama entidad relación definido es el siguiente:

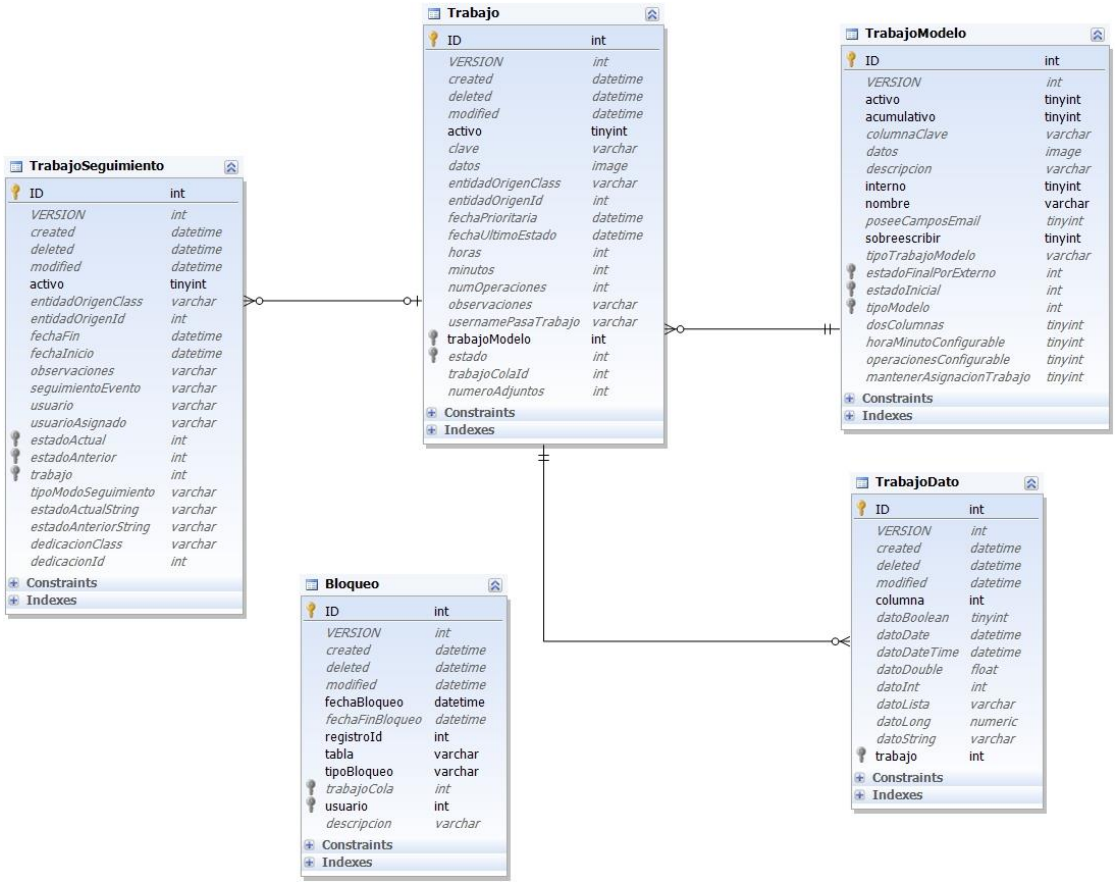

Figura 33. Diagrama entidad relación de tablas definidas para "Gestión de tareas"

3.5.5.2 Acceso a datos para la gestión y carga de tareas

A continuación se describen las clases encargadas del acceso a la base de datos, tanto para la consulta de datos como para la inserción, modificación o borrado de las tareas:

- La clase *DAOTrabajo.java* ofrece una serie de métodos la inserción, modificación y eliminación en base de datos las tareas.

- La clase *DAOTrabajoSeguimiento.java* nos ofrece una serie de métodos necesarios para el registro en base de datos de las acciones realizadas sobre una tarea.

- La clase *DAOTrabajoDato.java* nos ofrece una serie de métodos necesarios para el registro en base de datos de los valores de los campos de una tarea.

Los métodos más importantes de la clase *DAOTrabajo.java* (capa de acceso a datos) son los siguientes:

createTrabajo()
Método que crea un objeto Trabajo a crear en la base de datos.
findBy(TrabajoSpecification trabajoSpecification)
Método que recupera de la base de datos todos los trabajos que cumplen los criterios de filtrado indicados en el objeto *trabajoSpecification*.
findByModelo(TrabajoModelo trabajoModelo)
Método que recupera de la base de datos todos los trabajos cuyo modelo de trabajo coincide con el modelo de trabajo dado.
getNextTrabajoBy(TrabajoSpecification trabajoSpecification)
Método que recupera de la base de datos el primer trabajo no bloqueado que cumple con los criterios de búsqueda y ordenación que se indican en el objeto *trabajoSpecification*.
saveDatosTrabajo(Trabajo trabajo)
Método que actualiza en la base de datos los valores de los campos de un trabajo dado.
guardarEstado(Trabajo trabajo, TrabajoModeloEstado trabajoModeloEstado)
Método que actualiza en la base de datos el estado de un trabajo dado.

Tabla 39. Métodos más importantes de la clase DAOTrabajo.java

Los métodos más importantes de la clase *DAOTrabajoSeguimiento.java* (capa de acceso a datos) son los siguientes:

createTrabajoSegimiento()
Método que crea un nuevo objeto TrabajoSeguimiento a crear en la base de datos.
findBy(Trabajo trabajo)
Método que recupera de la base de datos todos los seguimientos de una tarea dada.
saveTrabajoSeguimiento(TrabajoSeguimiento trabajoSeguimiento)
Método que actualiza en la base de datos un seguimiento dado.

Tabla 40. Métodos más importantes de la clase DAOTrabajoSeguimiento.java

Los métodos más importantes de la clase *DAOTrabajoDato.java* (capa de acceso a datos) son los siguientes:

createTrabajoDato()
Método que crea un nuevo objeto TrabajoDato a crear en la base de datos.
findBy(Trabajo trabajo)
Método que recupera de la base de datos todos los datos de una tarea dada.
saveTrabajoDato(TrabajoDato trabajoDato)
Método que actualiza en la base de datos un dato dado.
deleteTrabajoDato(TrabajoDato trabajoDato)
Método que elimina de la base de datos un *TrabajoDato* dado.

Tabla 41. Métodos más importantes de la clase DAOTrabajoDato.java

3.5.5.3 Servicio de gestión y carga de tareas

Los servicios donde se define la lógica de negocio para la administración de los modelos de trabajo son:

- La clase *ServicioTrabajo.java* ofrece una serie de métodos para el alta, modificación y eliminación de tareas.
- La clase *ServicioTrabajoSeguimiento.java* nos ofrece una serie de métodos necesarios para la gestión de las acciones realizadas sobre una tarea.
- La clase *ServicioTrabajoDato.java* nos ofrece una serie de métodos necesarios para la gestión de los valores de los campos de una tarea.

Los métodos más importantes de la clase *ServicioTrabajo.java* (capa de lógica de negocio de la aplicación) son los siguientes:

createTrabajo()
Método que obtiene nuevo objeto *Trabajo*.
findBy(TrabajoSpecification trabajoSpecification)
Método que obtiene todos los trabajos/tareas que coinciden con los criterios de filtrado indicados en el objeto *trabajoSpecification*.
findByModelo(TrabajoModelo trabajoModelo)
Método que obtiene todos los trabajos de un modelo de trabajo dado.
txGuardarDatosTrabajo(Trabajo trabajo)
Método que envía a la capa de acceso a datos el objeto *Trabajo* preparado para ser registrado.
txActivarTrabajo(Trabajo trabajo)
Método que activa un trabajo.

txDesactivarTrabajo (Trabajo trabajo)
Método que desactiva un trabajo.
txAsignarTrabajo(Trabajo trabajo, Usuario usuario)
Método que realiza la asignación de un trabajo dado al usuario indicado.
txDesasignarTrabajo(Trabajo trabajo)
Método que realiza la desasignación de un trabajo dado que previamente estaba asignado.
txGetNextBy(TrabajoSpecification trabajoSpecification)
Método que devuelve el siguiente trabajo no bloqueado que cumple con los criterios de búsqueda y ordenación que se indican en el objeto *trabajoSpecification*.
txCargarTrabajo (TrabajoModelo trabajoModelo, File excel)
Método que crea trabajos de un modelo de trabajo desde un fichero excel.

Tabla 42. Métodos más importantes de la clase ServicioTrabajo.java

Los métodos más importantes de la clase *ServicioTrabajoSeguimiento.java* (capa de lógica de negocio de la aplicación) son los siguientes:

createTrabajoSeguimiento()
Método que obtiene nuevo objeto *TrabajoSeguimiento*.
findBy(Trabajo trabajo)
Método que obtiene todos los seguimientos de un trabajo dado.
saveTrabajoSeguimiento(TrabajoSeguimiento trabajoSeguimiento)
Método que registra un seguimiento dado.

Tabla 43. Métodos más importantes de la clase ServicioTrabajoSeguimiento.java

Los métodos más importantes de la clase *ServicioTrabajoDato.java* (capa de lógica de negocio de la aplicación) son los siguientes:

createTrabajoDato()
Método que obtiene un nuevo objeto *TrabajoDato*.
findBy(Trabajo trabajo)
Método que obtiene todos los datos de un trabajo/tarea dada.
saveTrabajoDato(TrabajoDato trabajoDato)
Método que registra un *TrabajoDato* dado.
deleteTrabajoDato(TrabajoDato trabajoDato)
Método que elimina un *TrabajoDato* dado.

Tabla 44. Métodos más importantes de la clase ServicioTrabajoDato.java

3.5.5.4 Diseño de pantalla de carga de tareas

La capa de presentación de la funcionalidad "Carga de tareas" está compuesta por la clase *GestionTrabajoActionBean.java* y las vistas *gestionModeloTrabajo.xhtml*.

La clase *GestionTrabajoActionBean.java* gestionará el fichero Excel donde vendrán definidas todas las tareas a cargar con los valores de los campos. Cada fila del fichero excel será una nueva a cargar. Cada columna indicará el valor del campo correspondiente.

Antes de realizar la carga de tareas se deberá configurar el modelo de trabajo indicando si existe alguna columna clave que identifique cada tarea unívocamente. Además se puede indicar si la carga será acumulativa o no. En el caso de ser acumulativa se tendrá en cuenta si la tarea ya ha sido registrada anteriormente en el sistema comparando con la columna clave. Se tendrá que indicar en este caso si se desea que la tarea sea sobrescrita o no.

A continuación se muestra el aspecto visual de la pestaña de configuración de carga de tareas:

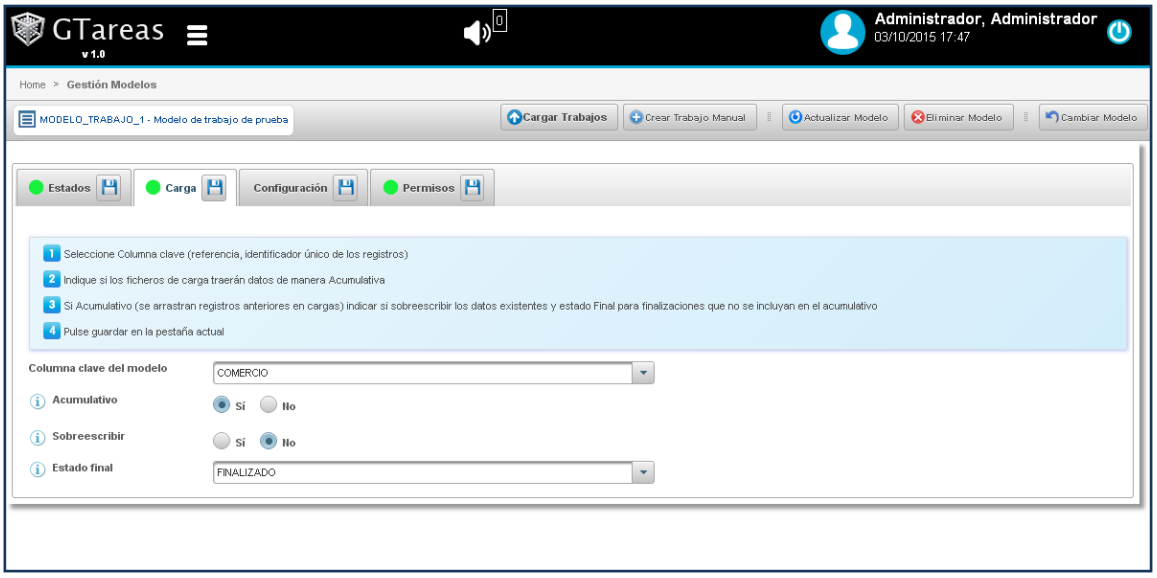

Figura 34. Aspecto visual de la pantalla "configuración de carga de tareas"

Una vez configurado el modelo de trabajo en cuanto a la carga de tareas, desde esta misma pantalla existe la opción de cargar el fichero Excel con las tareas a registrar. Al seleccionar la opción de "Cargar trabajos" seleccionaremos el fichero Excel a cargar y se registrarán las tareas que contiene el fichero en el sistema:

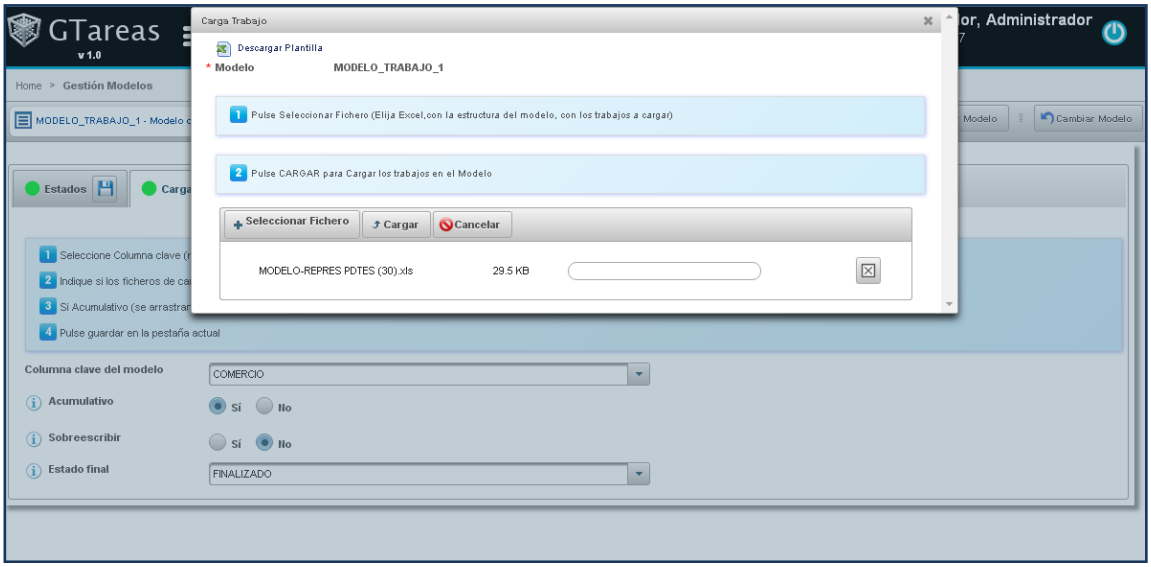

Figura 35. Aspecto visual de la pantalla "cargar tareas"

3.5.6 Implementación del requisito PB-0-009

En el punto 3.4.7 describimos las tareas necesarias para la implementación del requisito PB-0-009. Estas tareas son las siguientes:

- Diseño de esquema de base de datos con las tablas necesarias para el registro de la configuración de acceso a buzones de correo.
- Capa de acceso a base de datos para el registro de buzones, tareas y permisos.
- Servicio de la capa de negocio que se encargará de la administración de los buzones, registro de tareas y administración de permisos.
- Diseño de pantalla para la administración de buzones de correo. Además de la posibilidad de crear reglas de registro de tareas a partir de los correos y la posibilidad de administrar los permisos de los buzones.

3.5.6.1 Diseño de esquema de base de datos

En primer lugar definiremos las tablas necesarias para el registro de la configuración de acceso a buzones de correo. En la definición de las tablas se indicará el nombre de la tabla y los campos con su tipo de dato correspondiente.

En este caso se han definido las tablas: *Buzon*, *MensajeBuzon* y *MensajeAdjunto:*

- La tabla *Buzon* registrará los datos de la configuración de acceso a los buzones de correo.
- La tabla *MensajeBuzon* registrará todos los datos de los mensajes que se reciben en los buzones configurados.

- La tabla *MensajeAdjunto* registrará los datos de los ficheros adjuntos que puedan contener los mensajes recibidos en los buzones.

- La tabla *ReglaBuzon* registrará la configuración de las reglas con la información necesaria para crear tareas a partir de los mensajes recibidos.

El diagrama entidad relación definido es el siguiente:

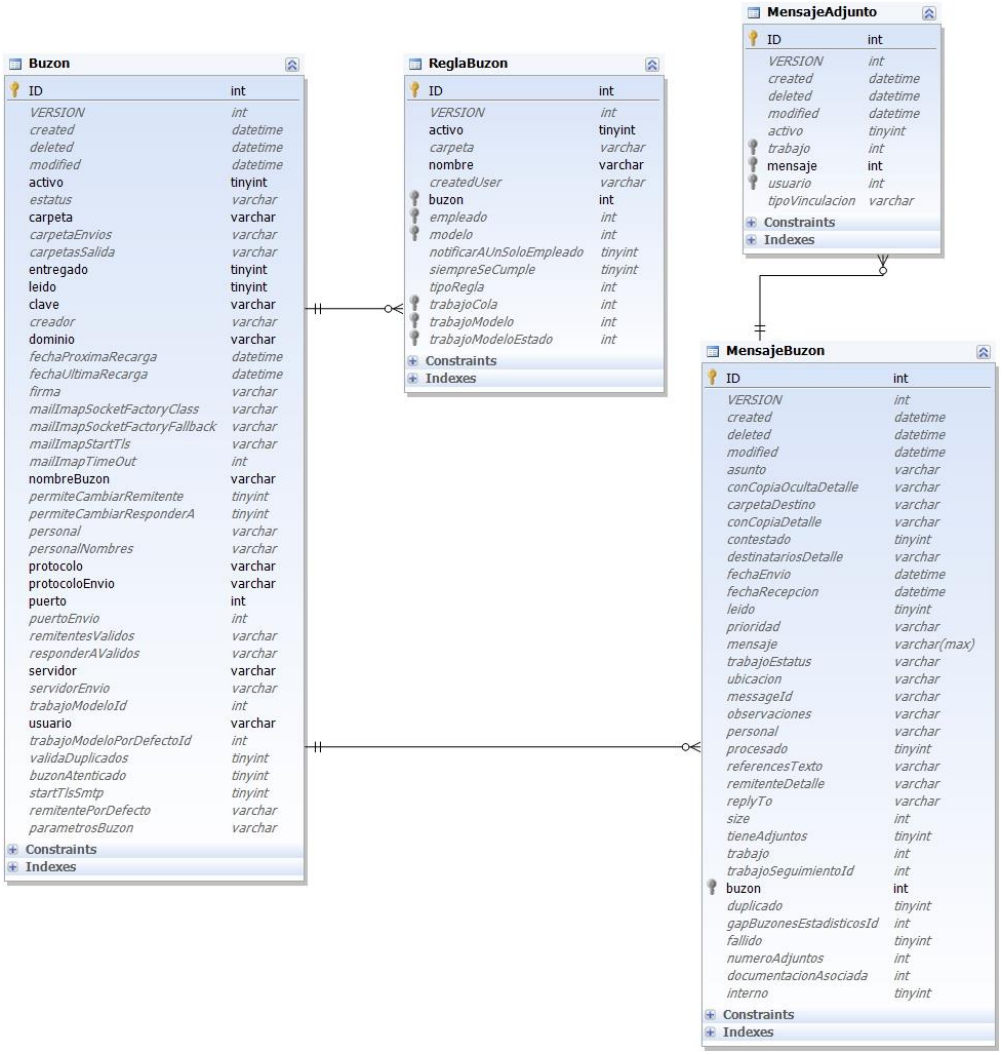

Figura 36. Diagrama entidad relación de tablas definidas para "Administración de buzones"

3.5.6.2 Acceso a datos para la administración de buzones y mensajes

A continuación se describen las clases encargadas del acceso a la base de datos, tanto para la consulta de datos como para la inserción, modificación o borrado de las tareas:

- La clase *DAOBuzon.java* ofrece una serie de métodos la inserción, modificación y eliminación en base de datos de buzones de correo.

- La clase *DAOMensajeBuzon.java* nos ofrece una serie de métodos necesarios para el registro en base de datos de mensajes de correo recibidos en cada uno de los buzones configurados.

- La clase *DAOMensajeAdjunto.java* nos ofrece una serie de métodos necesarios para el registro en base de datos de ficheros adjuntos dentro de los mensajes de correo recibidos.

- La clase *DAOReglaBuzon* nos ofrece una serie de métodos necesarios para el registro en base de datos de las reglas con la información necesaria para crear tareas a partir de los mensajes recibidos

Los métodos más importantes de la clase *DAOBuzon.java* (capa de acceso a datos) son los siguientes:

createBuzon()
Método que crea un nuevo objeto *Buzon* a crear en la base de datos.
findBy(BuzonSpecification buzonSpecification)
Método que recupera de la base de datos todos los buzones que cumplen con los criterios de filtrado indicados en el objeto *buzonSpecification*.
saveOrUpdateBuzon(Buzon buzon)
Método que actualiza en la base de datos un buzón dado.
deleteBuzon(Buzon buzon)
Método que elimina de la base de datos un buzón dado.

Tabla 45. Métodos más importantes de la clase DAOBuzon.java

Los métodos más importantes de la clase *DAOMensajeBuzon.java* (capa de acceso a datos) son los siguientes:

createMensajeBuzon()
Método que crea un nuevo objeto *MensajeBuzon* a crear en la base de datos.
findAllBy(Buzon buzon)
Método que recupera de la base de datos todos los mensajes registrados pertenecientes a un buzón dado.
saveOrUpdateMensajeBuzon(MensajeBuzon mensajeBuzon)
Método que actualiza en la base de datos un mensaje dado.
deleteMensajeBuzon(MensajeBuzon mensajeBuzon)
Método que elimina de la base de datos un mensaje dado.

Tabla 46. Métodos más importantes de la clase DAOMensajeBuzon.java

Los métodos más importantes de la clase *DAOMensajeAdjunto.java* (capa de acceso a datos) son los siguientes:

createMensajeAdjunto()
Método que crea un nuevo objeto *MensajeAdjunto* a crear en la base de datos.
findAllBy(MensajeBuzon mensajeBuzon)
Método que recupera de la base de datos todos los adjuntos registrados pertenecientes a un mensaje dado.
saveOrUpdateMensajeAdjunto(MensajeAdjunto mensajeAdjunto)
Método que actualiza en la base de datos un adjunto dado.
deleteMensajeAdjunto(MensajeAdjunto mensajeAdjunto)
Método que elimina de la base de datos un adjunto dado.

Tabla 47. Métodos más importantes de la clase DAOMensajeAdjunto.java

Los métodos más importantes de la clase *DAOReglaBuzon.java* (capa de acceso a datos) son los siguientes:

createReglaBuzon()
Método que crea un nuevo objeto *ReglaBuzon* a crear en la base de datos.
findAllBy(Buzon Buzon)
Método que recupera de la base de datos todas las reglas pertenecientes a un buzón dado.
saveOrUpdateRegla(ReglaBuzon reglaBuzon)
Método que actualiza en la base de datos una regla dada.
deleteReglaBuzon(ReglaBuzon reglaBuzon)
Método que elimina de la base de datos una regla dada.

Tabla 48. Métodos más importantes de la clase DAOReglaBuzon.java

3.5.6.3 Servicio de administración de buzones y mensajes

Los servicios donde se define la lógica de negocio para la administración de los buzones y sus mensajes son:

- La clase *ServicioBuzon.java* ofrece una serie de métodos para el alta, modificación y eliminación de buzón. Además ofrece una serie de métodos necesarios para la lectura y posterior carga de mensajes desde el buzón físico de correo.

- La clase ServicioMensajeBuzon.java nos ofrece una serie de métodos necesarios para la gestión de mensajes de correo. Registro de los mensajes recibidos y la aplicación de las reglas para la creación de tareas. Además se incluyen en este servicio los métodos necesarios para la gestión de los adjuntos a los mensajes.

Los métodos más importantes de la clase *ServicioBuzon.java* (capa de lógica de negocio de la aplicación) son los siguientes los siguientes:

createBuzon() Método que obtiene un nuevo objeto *Buzon*.
findBy(BuzonSpecification buzonSpecification) Método que obtiene todos los buzones que cumplen con los criterios de filtrado indicados en el objeto *buzonSpecification*.
saveOrUpdateBuzon(Buzon buzon) Método que registra un buzón dado.
deleteBuzon(Buzon buzon) Método que elimina un buzón dado.

Tabla 49. Métodos más importantes de la clase ServicioBuzon.java

Los métodos más importantes de la clase *ServicioMensajeBuzon.java* (capa de lógica de negocio de la aplicación) son los siguientes los siguientes:

createMensajeBuzon() Método que obtiene un nuevo objeto Mensaje*Buzon*.
findBy(BuzonSpecification buzonSpecification) Método que obtiene todos los buzones que cumplen con los criterios de filtrado indicados en el objeto *buzonSpecification*.
txRegistrarMensaje(MensajeBuzon mensajeBuzon) Método que registra un mensaje dado.
txRegistrarMensajeConAdjuntos(MensajeBuzon mensajeBuzon) Método que registra un mensaje con adjuntos dado.
saveOrUpdateReglaBuzon(ReglaBuzon reglaBuzon) Método que registra una regla dada.
recargarMensajes(Buzon buzon) Método que recarga los mensajes de un buzon dado. Se conecta al buzon indicado y registra todos aquellos mensajes pendientes.
txAplicarReglas(MensajeBuzon mensajeBuzon) Método que aplica las reglas correspondientes al mensaje dado. Dependiendo de las características del mensaje y del buzon del que procede se le aplicarán las reglas pertinentes.

Tabla 50. Métodos más importantes de la clase ServicioMensajeBuzon.java

3.5.6.4 Diseño de pantalla de administración de buzones

La capa de presentación de la funcionalidad "Administración de buzones" está compuesta por las clases *ListadoBuzonActionBean.java*, *GestionBuzonActionBean.java* y las vistas *listadoBuzon.xhtml*, *gestionBuzonTrabajo.xhtml*.

La clase *ListadoBuzonActionBean.java* gestionará los métodos necesarios para obtener la lista de los buzones existentes en el sistema. La clase *GestionBuzonActionBean.java* ofrecerá los métodos para dar de alta un nuevo buzón o modificar o eliminar alguno de los buzones existentes.

La vista *listadoBuzon.xhtml*, encargada de visualizar el listado de los buzones existentes, mostrará la siguiente pantalla:

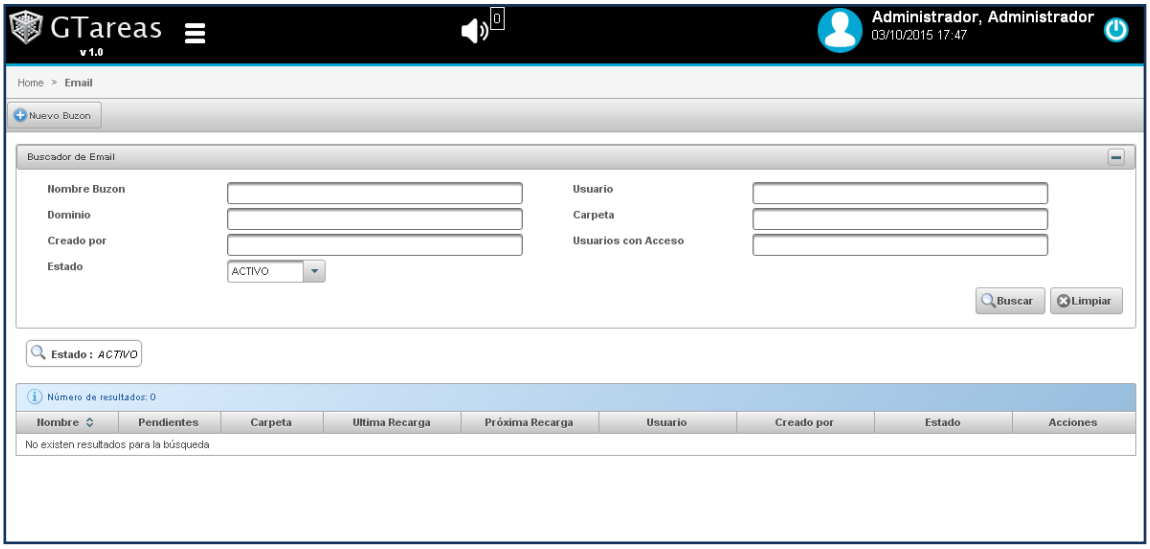

Figura 37. Aspecto visual de la pantalla "listado de buzones"

Al seleccionar la opción de "Nuevo buzón" entraremos en la vista *gestionBuzon.xhtml* que tendrá el siguiente aspecto:

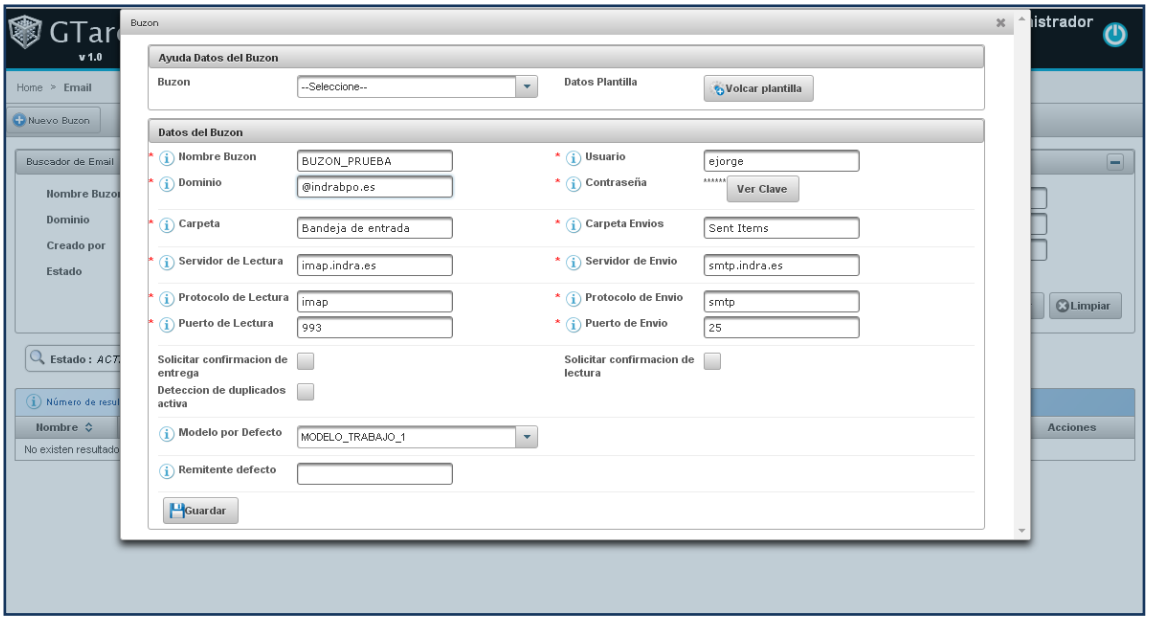

Figura 38. Aspecto visual de la pantalla "gestión de buzones"

Una vez creada la configuración del buzón de trabajo se podrán gestionar las reglas a aplicar a los mensajes que se reciban. La siguiente pantalla muestra el alta de una regla para los mensajes recibidos:

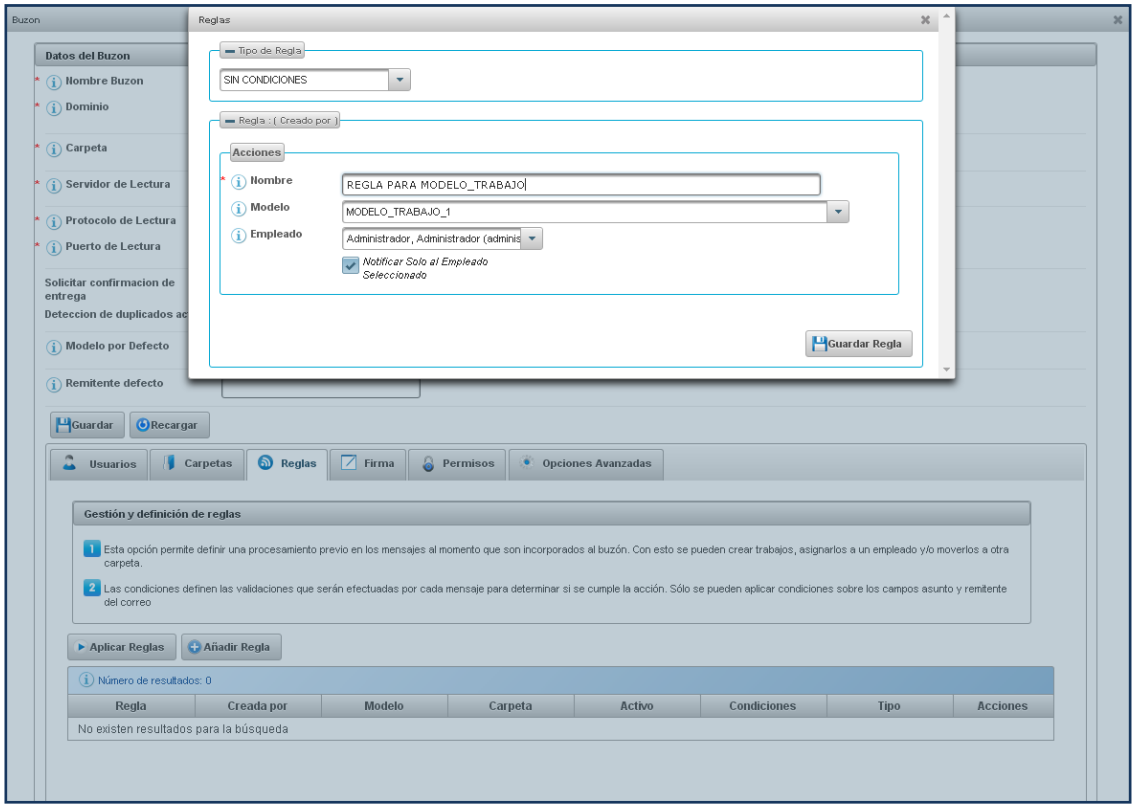

Figura 39. Aspecto visual de la pantalla "alta de regla para mensajes recibidos"

Una vez creada la regla, está se aplicará en el momento que se realice la conexión al buzón y se reciban nuevos mensajes en dicho buzón. Cada mensaje recibido creará una tarea del modelo de trabajo seleccionado en la creación de la regla.

Desde la pantalla de gestión del buzón, se podrán administrar los permisos de acceso al buzón. La pantalla mostrará lo siguiente:

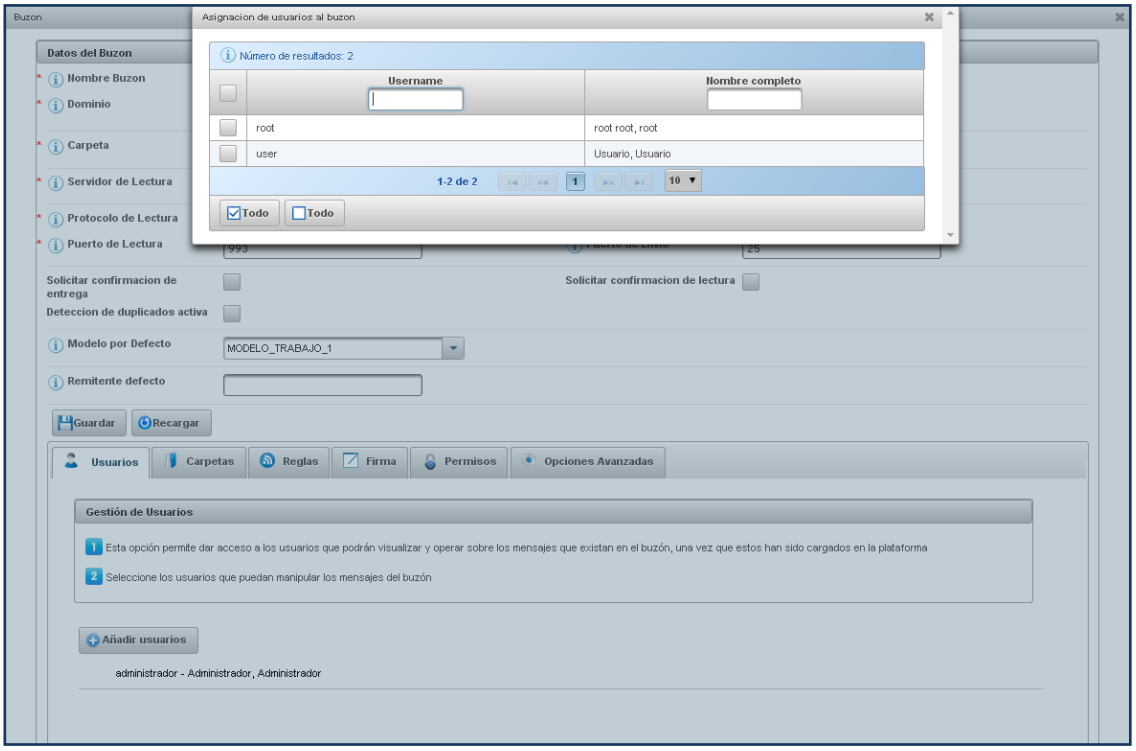

Figura 40. Aspecto visual de la pantalla "permisos del buzón"

3.5.7 Implementación del requisito PB-0-010

En el punto 3.4.8 describimos las tareas necesarias para la implementación del requisito PB-0-010. Estas tareas son las siguientes:

- Diseño de esquema de base de datos con las tablas necesarias para el registro de colas de trabajo y almacenar criterios de filtrado y ordenación.
- Capa de acceso a base de datos para el registro de los datos de las colas de trabajo y sus criterios de filtrado y ordenación.
- Servicio de la capa de negocio que se encargará de la administración de las colas de trabajo y de gestionar y tratar sus criterios de filtrado y ordenación.
- Diseño de pantalla para la administración de colas de trabajo. Además de la posibilidad de configurar las colas mediante criterios de filtrado y criterios de ordenación.

3.5.7.1 Diseño de esquema de base de datos

En primer lugar definiremos las tablas necesarias para el registro de colas de trabajo y su configuración.

En este caso se han definido las tablas: *TrabajoCola* y *TrabajoColaUsuario:*

- La tabla *TrabajoCola* registrará los datos de la configuración de las colas de trabajo, sus criterios de filtrado y sus criterios de ordenación.

- La tabla *TrabajoColaUsuario* registrará todos los usuarios con permisos tanto de administrador dentro de las colas de trabajo, como de actuación dentro de las tareas servidas por las colas de trabajo.

El diagrama entidad relación definido es el siguiente:

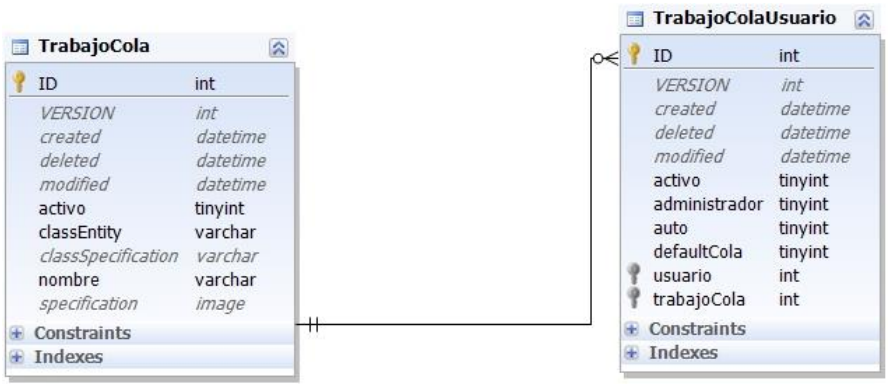

Figura 41. Diagrama entidad relación de tablas definidas para "Administración de colas de trabajo"

3.5.7.2 Acceso a datos para la administración de colas de trabajo

A continuación se describen las clases encargadas del acceso a la base de datos, tanto para la consulta de las colas de trabajo existentes como para la inserción, modificación o borrado de colas de trabajo:

- La clase *DAOTrabajoCola.java* ofrece una serie de métodos la inserción, modificación y eliminación en base de datos de colas de trabajo.

- La clase *DAOTrabajoColaUsuario.java* nos ofrece una serie de métodos necesarios para el registro en base de datos de los permisos de los usuarios dentro de las colas de trabajo.

Los métodos más importantes de la clase *DAOTrabajoCola.java* (capa de acceso a datos) son los siguientes:

createTrabajoCola()
Método que crea un nuevo objeto *TrabajoCola* a crear en la base de datos.
findBy(TrabajoModelo trabajoModelo)
Método que recupera de la base de datos todas las colas de trabajo definidas para el modelo de trabajo dado.
saveOrUpdateTrabajoCola(TrabajoCola trabajoCola)
Método que actualiza en la base de datos una cola de trabajo.
deleteTrabajoCola(TrabajoCola trabajoCola)

Método que elimina de la base de datos una cola de trabajo.

Tabla 51. Métodos más importantes de la clase DAOTrabajoCola.java

Los métodos más importantes de la clase *DAOTrabajoColaUsuario.java* (capa de acceso a datos) son los siguientes:

findAllUsuarioBy(TrabajoCola TrabajoCola)
Método que recupera de la base de datos todos los usuarios dentro de una cola de trabajo dada.
saveOrUpdateUsuario(TrabajoCola TrabajoCola, Usuario usuario)
Método que actualiza en la base de datos un usuario dado dentro de una cola de trabajo dada.
deleteUsuario(TrabajoCola TrabajoCola, Usuario usuario)
Método que elimina de la base de datos un usuario dado dentro de una cola de trabajo dada.

Tabla 52. Métodos más importantes de la clase DAOTrabajoColaUsuario.java

3.5.7.3 Servicio de administración de colas de trabajo

Los servicios donde se define la lógica de negocio para la administración de las colas de trabajo son:

- La clase *ServicioTrabajoCola.java* ofrece una serie de métodos para el alta, modificación y eliminación de colas de trabajo. Además ofrece una serie de métodos necesarios para la definición de los criterios de búsqueda y ordenación que definirán el modo de trabajo de las colas.

- La clase *ServicioTrabajoColaUsuario.java* nos ofrece una serie de métodos necesarios para la gestión de los permisos dentro de las colas de trabajo.

Los métodos más importantes de la clase *ServicioTrabajoCola.java* (capa de lógica de negocio de la aplicación) son los siguientes los siguientes:

createTrabajoCola()
Método que obtiene un nuevo objeto *TrabajoCola*.
findTrabajoColaBy(TrabajoModelo trabajoModelo)
Método que obtiene todas las colas de trabajo definidas para el modelo de trabajo dado.
saveOrUpdateTrabajoCola(TrabajoCola trabajoCola)
Método que registra una cola de trabajo.
deleteTrabajoCola(TrabajoCola trabajoCola)
Método que elimina una cola de trabajo.
txGetNextTrabajoNoBloqueado(TrabajoCola trabajoCola, Usuario usuario)
Método que obtiene la siguiente tarea no bloqueda dentro de una cola de trabajo dada para un usuario dado.
txCrearColasEstadosByModelo(TrabajoModelo trabajoModelo)

Método que crea tantas colas de trabajo como estados no finales tenga un modelo de trabajo dado.

Tabla 53. Métodos más importantes de la clase ServicioTrabajoCola.java

Los métodos más importantes de la clase *ServicioTrabajoColaUsuario.java* (capa de lógica de negocio de la aplicación) son los siguientes:

findAllUsuarioBy(TrabajoCola TrabajoCola)
Método que obtiene todos los usuarios dentro de una cola de trabajo dada.
saveOrUpdateUsuario(TrabajoCola TrabajoCola, Usuario usuario)
Método que registra un usuario dado dentro de una cola de trabajo dada.
deleteUsuario(TrabajoCola TrabajoCola, Usuario usuario)
Método que elimina un usuario dado dentro de una cola de trabajo dada.

Tabla 54. Métodos más importantes de la clase ServicioTrabajoColaUsuario.java

3.5.7.4 Diseño de pantalla de administración de colas de trabajo

La capa de presentación de la funcionalidad "Administración de colas de trabajo" está compuesta por las clases *ListadoTrabajoColaActionBean.java* y *GestionTrabajoCola.java* y las vistas *listadoTrabajoCola.xhtml* y *gestionTrabajoCola.xhtml*.

La clase *ListadoTrabajoColaActionBean.java* gestionará los métodos necesarios para obtener la lista de las colas de trabajo existentes en el sistema. La clase *GestionTrabajoCola.java* ofrecerá los métodos para dar de alta una nueva cola de trabajo o modificar o eliminar alguna de las colas de trabajo existentes.

La vista *listadoTrabajoCola.xhtml*, encargada de visualizar el listado de las colas de trabajo existentes para un determinado modelo de trabajo, mostrará la siguiente pantalla:

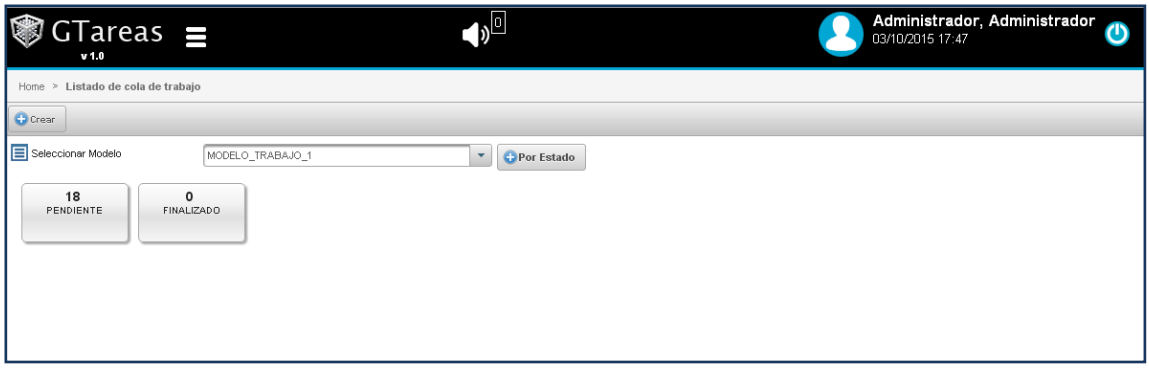

Figura 42. Aspecto visual de la pantalla "listado de colas de trabajo"

Observamos en la figura anterior que el listado de colas de trabajo para el modelo seleccionado está vacío, no existe ninguna cola de trabajo creada. Para crear una nueva cola de trabajo el sistema da las opciones siguientes: Crear nueva cola de trabajo o crear tantas colas de trabajo como estados no finales tenga el modelo de trabajo.

Al escoger la opción de "Crear" nueva cola de trabajo, se mostrará la siguiente pantalla de alta:

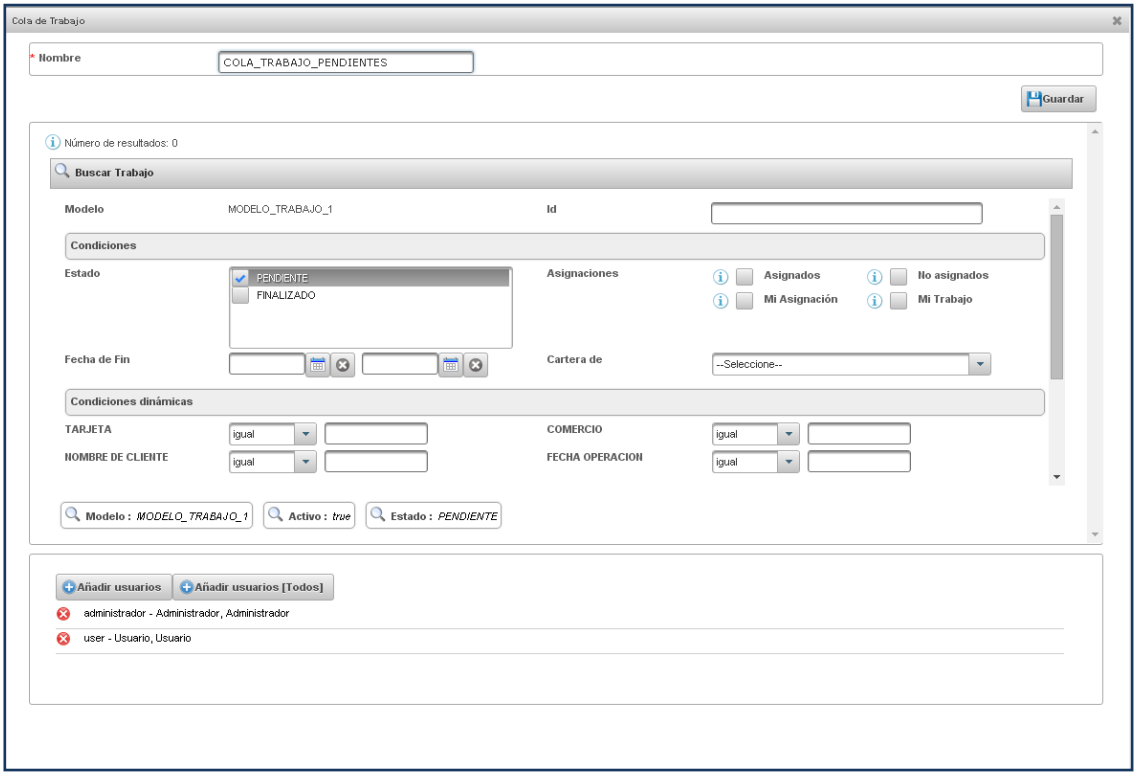

Figura 43. Aspecto visual de la pantalla "Nueva cola de trabajo"

Observamos en la figura anterior la cola de trabajo creada. En este caso, los criterios de filtrado son todas aquellas tareas del modelo de trabajo seleccionado en estado "PENDIENTE". Se han dado permisos a dos usuarios dentro de la cola de trabajo.

Una vez creada la cola de trabajo observamos cómo queda el listado de colas de trabajo para el modelo seleccionado en la siguiente pantalla:

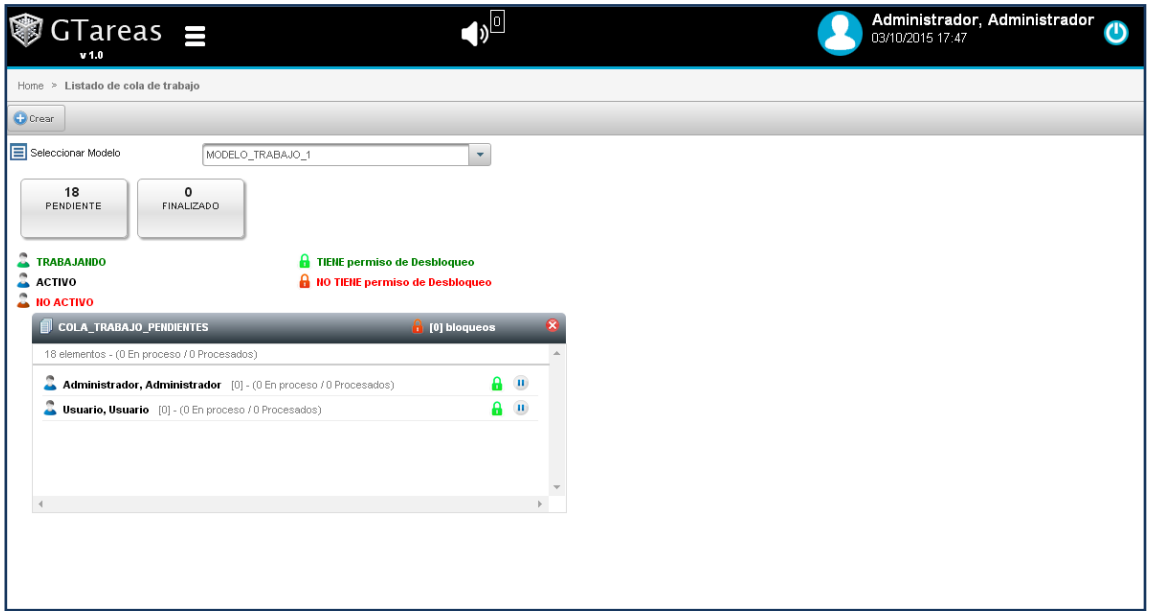

Figura 44. Aspecto visual de la pantalla "Listado colas de trabajo"

3.5.8 Implementación del requisito PB-0-011

En el punto 3.4.9 describimos las tareas necesarias para la implementación del requisito PB-0-011. Estas tareas son las siguientes:

- Capa de acceso a base de datos para la consulta de tareas dentro de las colas de trabajo.
- Servicio de la capa de negocio que se encargará de recoger la información necesaria para el seguimiento de tareas en las colas de trabajo.
- Diseño de pantalla para la monitorización de las colas de trabajo dentro de un modelo de trabajo.

3.5.8.1 Acceso a datos para la monitorización de tareas

A continuación se describen las clases encargadas del acceso a la base de datos para las consultas necesarias en la monitorización del estado de colas de trabajo:

- La clase *DAOTrabajo.java* ofrece una serie de métodos de consulta para los trabajos/tareas dentro de las colas de trabajo.

Los métodos más utilizados de la clase *DAOTrabajo.java* (capa de acceso a datos) para la monitorización de modelos de trabajo:

findTrabajoBy(TrabajoCola trabajoCola)
Método que consulta en la base de datos todos los trabajos que cumplan los criterios de búsqueda indicados en la cola de trabajo.

getCountBy(TrabajoCola trabajoCola) Método que recupera de la base de datos el número de trabajos que cumplan los criterios de búsqueda indicados en la cola de trabajo.
findTrabajoByEstado(TrabajoModelo trabajoModelo) Método que recupera de la base de datos todos los trabajos de un modelo de trabajo agrupados por estado.

Tabla 55. Métodos utilizados de la clase DAOTrabajo.java para la monitorización

3.5.8.2 Servicio de monitorización de tareas

Los servicios donde se define la lógica de negocio para la monitorización de modelos de trabajo son:

- La clase *ServicioTrabajo.java* ofrece una serie de métodos para monitorización de los trabajos dentro de un modelo de trabajo, contemplando las colas de trabajo configuradas.

Los métodos usados de la clase *ServicioTrabajo.java* (capa de lógica de negocio de la aplicación) para la monitorización de modelos de trabajo, son los siguientes

findTrabajoBy(TrabajoCola trabajoCola) Método que obtiene todos los trabajos que cumplan los criterios de búsqueda indicados en la cola de trabajo.
getCountBy(TrabajoCola trabajoCola) Método que obtiene el número de trabajos que cumplan los criterios de búsqueda indicados en la cola de trabajo.
findTrabajoByEstado(TrabajoModelo trabajoModelo) Método que obtiene agrupados por estado todos los trabajos de un modelo de trabajo.

Tabla 56. Métodos utilizados de la clase DAOTrabajo.java para la monitorización

3.5.8.3 Diseño de pantalla de monitorización de tareas

La capa de presentación de la funcionalidad "Monitorización de modelos de trabajo" está compuesta por la clase *MonitorModeloTrabajoActionBean.java* y la vista *monitorModeloTrabajo.xhtml*.

La clase *MonitorModeloTrabajoActionBean.java* gestionará los métodos necesarios para obtener los datos de las tareas del modelo de trabajo seleccionado.

La vista *monitorModeloTrabajo.xhtml*, encargada de visualizar el monitor de la situación de las tareas del modelo de trabajo seleccionado y la situación de las colas de trabajo configuradas, mostrará la siguiente pantalla:

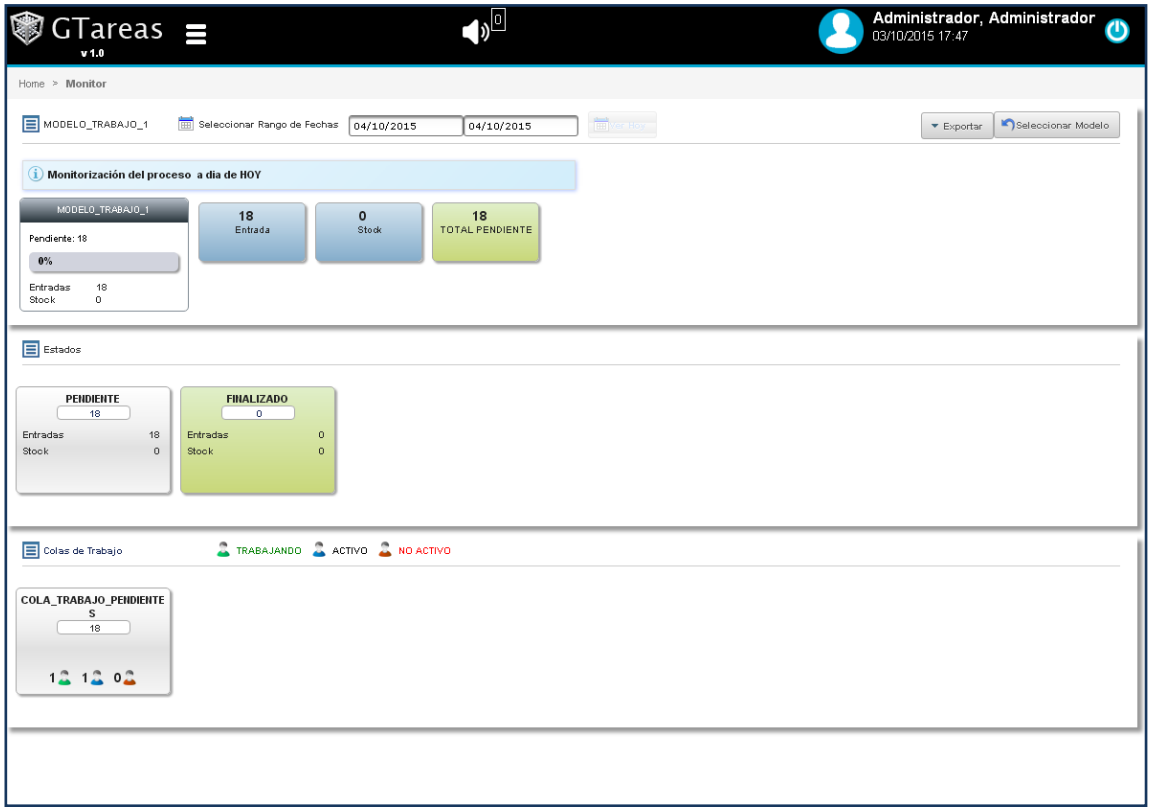

Figura 45. Aspecto visual de la pantalla "Monitor de modelo de trabajo"

Observamos en la figura anterior que el monitor de un modelo de trabajo seleccionado nos ofrece un resumen de la situación en la que se encuentran las tareas del modelo de trabajo, como se encuentran las tareas respecto a su estado y la situación de las colas de trabajo creadas.

3.5.9 Implementación del requisito PB-0-012

En el punto 3.4.10 describimos las tareas necesarias para la implementación del requisito PB-0-012. Estas tareas son las siguientes:

- Capa de acceso a base de datos para la consulta de tareas y acceso a sus datos.
- Servicio de la capa de negocio que se encargará de recoger los resultados de las búsquedas con los criterios definidos.
- Diseño de pantalla para la búsqueda de tareas mediante criterios de filtrado y mostrar resultados siguiendo criterios de ordenación. Posibilidad de acceso a los datos de las tareas resultantes.

3.5.9.1 Acceso a datos para la consulta y gestión de tareas

La clase utilizada para la consulta de tareas en la base de datos será *DAOTrabajo.java*. Esta clase y sus métodos ya ha sido descrita en el punto 3.5.5.2. En

esta clase se usarán los métodos para la consulta de las tareas en las que se aplican los filtros de búsqueda y ordenación. Estos métodos son los siguientes:

findBy(TrabajoSpecification trabajoSpecification)
Método que recupera de la base de datos todos los trabajos que cumplen los criterios de filtrado indicados en el objeto *trabajoSpecification*.
findByModelo(TrabajoModelo trabajoModelo)
Método que recupera de la base de datos todos los trabajos cuyo modelo de trabajo coincide con el modelo de trabajo dado.

Tabla 57. Métodos usados de la clase DAOTrabajo.java

3.5.9.2 Servicio de consulta y gestión de tareas

La clase utilizada para la capa de lógica de negocio en la consulta y gestión de tareas es *ServicioTrabajo.java*. Esta clase y sus métodos ya ha sido descrita en el punto 3.5.5.3. Los métodos usados de la clase *ServicioTrabajo.java* (capa de lógica de negocio de la aplicación) para la consulta y la gestión de tareas, son los siguientes:

findBy(TrabajoSpecification trabajoSpecification)
Método que obtiene todos los trabajos/tareas que coinciden con los criterios de filtrado indicados en el objeto *trabajoSpecification.*
findByModelo(TrabajoModelo trabajoModelo)
Método que obtiene todos los trabajos de un modelo de trabajo dado.

Tabla 58. Métodos usados de la clase ServicioTrabajo.java

3.5.9.3 Diseño de pantalla de consulta y acceso de tareas

La capa de presentación de la funcionalidad "Consulta y gestión de tareas" está compuesta por las clases *ListadoTrabajoActionBean.java* y *GestionTrabajoActionBean.java* y las vistas *listadoTrabajo.xhtml* y *gestionTrabajo.xhtml*.

La clase *ListadoTrabajoActionBean.java* gestionará los métodos necesarios para obtener los datos de las tareas que cumplen los criterios de búsqueda indicados por el usuario en el filtro de tareas de la pantalla.

La vista *listadoTrabajo.xhtml*, será encargada de visualizar el buscador de tareas, con dos partes diferenciadas: el filtro de búsqueda por un lado y el resultado de la búsqueda por otro. Se mostrará la siguiente pantalla:

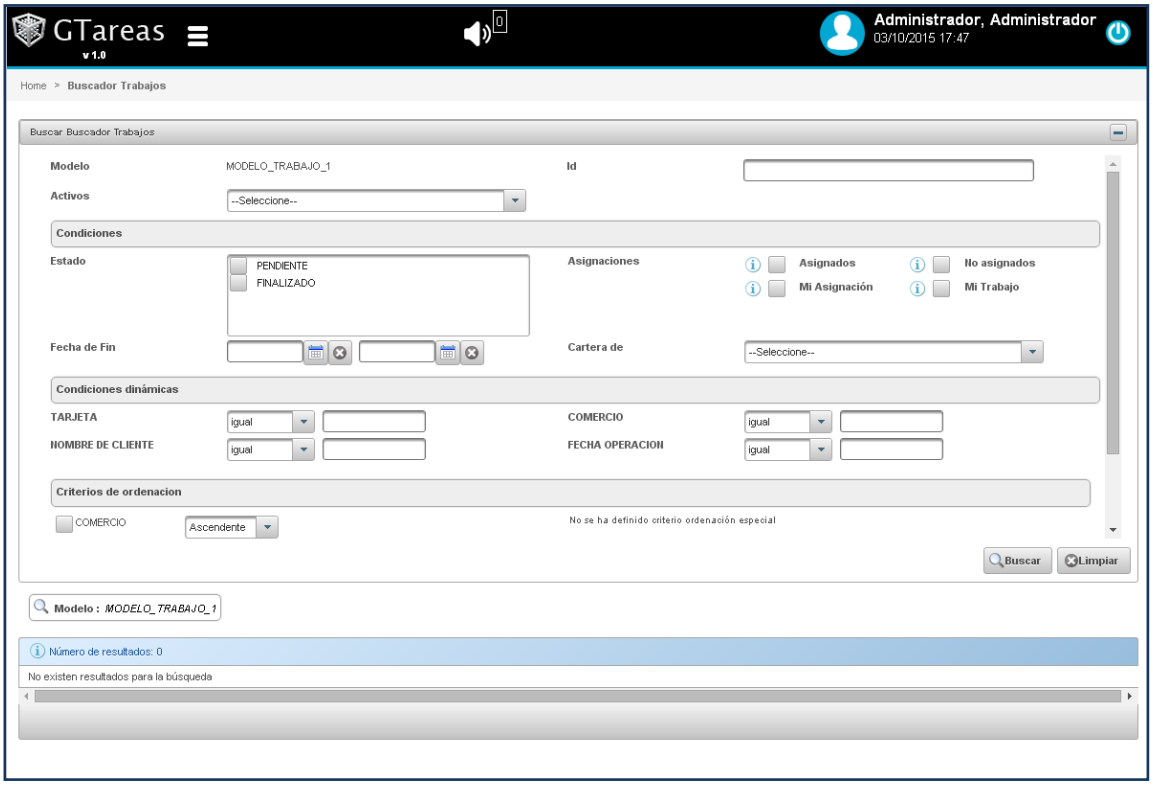

Figura 46. Aspecto visual de la pantalla "Buscador de tareas"

Observamos en la figura anterior que el buscador de tareas nos ofrece un filtro completo de tareas. Este filtro se compone de algunos campos de búsqueda generales (modelo, estado, etc.) y otros campos que se indican en la configuración del modelo de trabajo. Para agregar campos en el filtro de tareas desde la configuración del modelo de trabajo se debe marcar la opción de "filtro" en los campos que se desee que aparezcan.

El resultado de la búsqueda muestra aquellos campos que se ha indicado desde la configuración del modelo de trabajo. Para ver los campos en el resultado de búsqueda hay que marcar la opción "resultado" en los campos que se desee que aparezcan.

A continuación se muestra el resultado de la búsqueda después de pulsar el botón "buscar" sin indicar ningún valor en ningún campo del filtro:

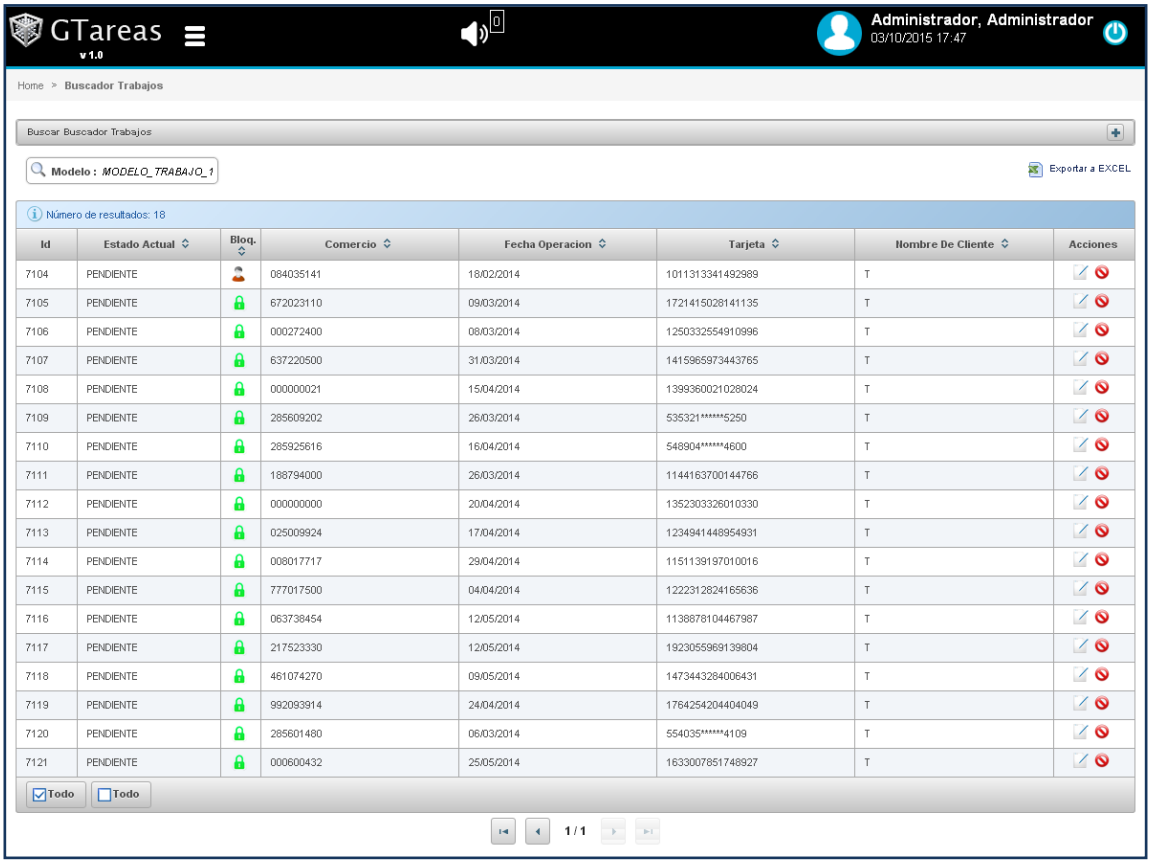

Figura 47. Aspecto visual de la pantalla "Resultado búsqueda de tareas"

Desde el la tabla del resultado de la búsqueda se podrá acceder al detalle de cada tarea para ver sus datos y hacer las gestiones pertinentes sobre ella. La pantalla de la gestión de la tarea será la siguiente:

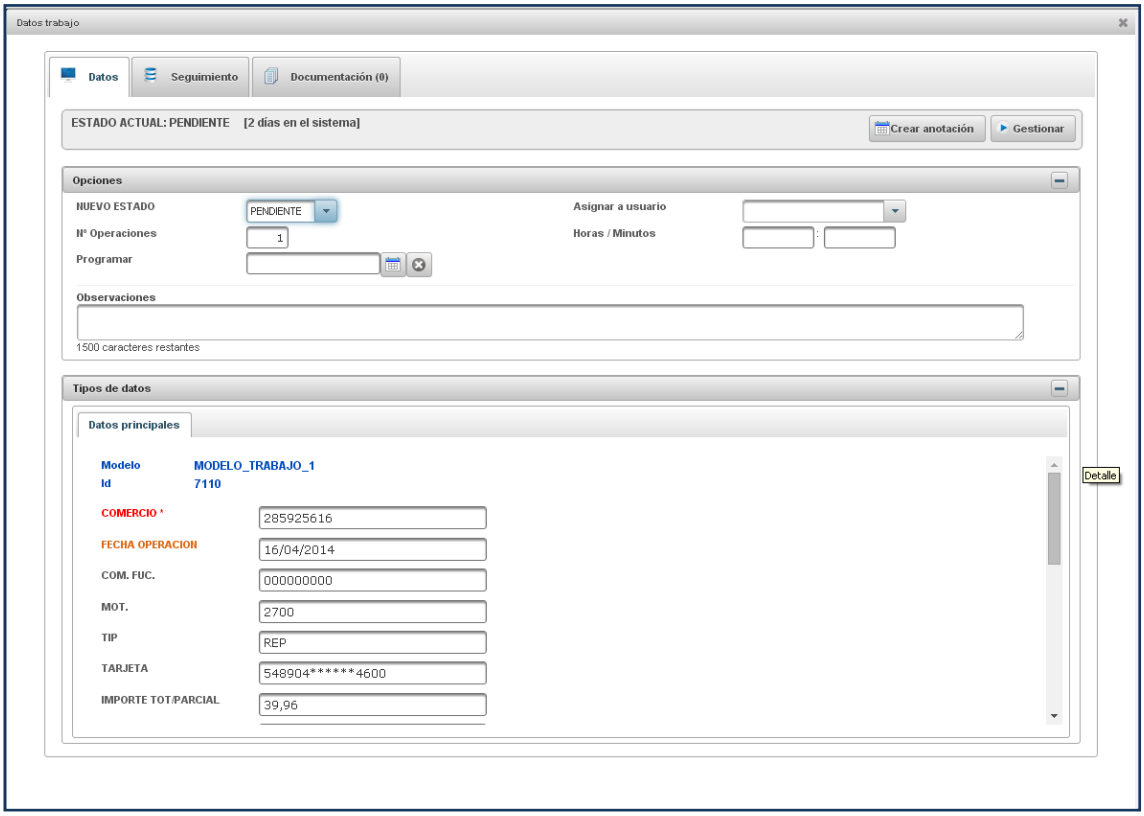

Figura 48. Aspecto visual de la pantalla "Detalle y gestión de tarea"

Observamos en la figura anterior toda la información mostrada de una tarea en concreto. Se indican los datos generales de la tarea y además los datos que han sido indicados desde la configuración del modelo de trabajo. Desde esta pantalla un usuario podrá gestionar la realización de la tarea realizando cambios de estado en ella. Además podrá indicar los valores pertinentes en los distintos campos de la tarea.

3.5.10 Implementación del requisito PB-0-013

En el punto 3.4.11 describimos las tareas necesarias para la implementación del requisito PB-0-013. Estas tareas son las siguientes:

- Capa de acceso a base de datos para la consulta de tareas dentro de las colas de trabajo.

- Servicio de la capa de negocio que se encargará de servir tareas de la cola de trabajo a la que se conecta el usuario siguiendo los criterios de filtrado y ordenación indicados en la cola de trabajo.

- Diseño de pantalla para la conexión de los usuarios a las colas de trabajo disponibles a las que tenga permisos.

3.5.10.1 Acceso a datos para el reparto de tareas

La clase utilizada para la consulta de tareas en la base de datos será *DAOTrabajo.java*. Esta clase y sus métodos ya ha sido descrita en el punto 3.5.5.2. En esta clase se usarán los métodos para la consulta de las tareas en las que se aplican los filtros de búsqueda y ordenación indicados en la cola de trabajo donde el usuario esté trabajando. Estos métodos son los siguientes:

getNextTrabajoBy(TrabajoSpecification trabajoSpecification) Método que recupera de la base de datos el primer trabajo no bloqueado que cumple con los criterios de búsqueda y ordenación que se indican en el objeto *trabajoSpecification*.
findBy(TrabajoSpecification trabajoSpecification) Método que recupera de la base de datos todos los trabajos que cumplen los criterios de filtrado indicados en el objeto *trabajoSpecification*.

Tabla 59. Métodos usados de la clase DAOTrabajo.java

3.5.10.2 Servicio de reparto de tareas

La clase utilizada para la capa de lógica de negocio en el reparto de tareas es *ServicioTrabajo.java*. Esta clase y sus métodos ya ha sido descrita en el punto 3.5.5.3. Los métodos usados de la clase *ServicioTrabajo.java* (capa de lógica de negocio de la aplicación) para el reparto de tareas, son los siguientes:

findBy(TrabajoSpecification trabajoSpecification) Método que obtiene todos los trabajos/tareas que coinciden con los criterios de filtrado indicados en el objeto *trabajoSpecification.*
findByModelo(TrabajoModelo trabajoModelo) Método que obtiene todos los trabajos de un modelo de trabajo dado.

Tabla 60. Métodos usados de la clase ServicioTrabajo.java

3.5.10.3 Diseño de pantalla de reparto de tareas

La capa de presentación de la funcionalidad "Reparto de tareas" está compuesta por la clase *RepartoTrabajoActionBean.java* y las vistas *repartoTrabajo.xhtml* y *gestionTrabajo.xhtml*.

La clase *RepartoTrabajoActionBean.java* gestionará los métodos necesarios para obtener los datos de las tareas que cumplen los criterios de búsqueda indicados en la cola de trabajo donde el usuario esté trabajando. Una vez obtenida la lista de tareas, el servicio irá sirviendo las tareas una a una al usuario para que realice las gestiones pertinentes.

La vista *repartoTrabajo.xhtml*, será encargada de visualizar las colas de trabajo disponibles para el usuario. El usuario podrá conectarse a cualquiera de las colas de trabajo en las que tenga permisos. Se mostrará la siguiente pantalla:

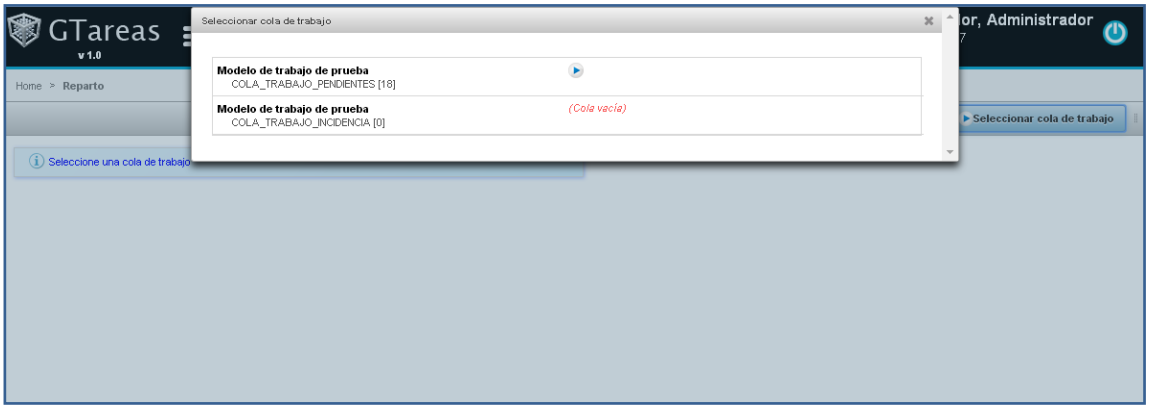

Figura 49. Aspecto visual de la pantalla "Conexión a cola de trabajo"

Observamos en la figura anterior el listado de colas de trabajo disponibles para el usuario. Para cada cola de trabajo se indicará el número de tareas que componen esa cola. En el caso de que el usuario no disponga de permisos para conectarse a ninguna cola de trabajo, la lista será vacía. En el caso de que el usuario disponga de permisos para solo una cola de trabajo, la conexión se hará automáticamente a esa cola de trabajo y comenzará a servirle la primera tarea disponible.

Una vez el usuario se conecte a la cola de trabajo seleccionada, el sistema comenzará a servirle una a una las tareas disponibles es ese momento. La pantalla de le gestión de la tarea será la misma que se si accediera desde el buscador de tareas. El usuario podrá realizar el cambio de estado pertinente dentro de la tarea, así como los cambios de los valores de los campos que considere oportuno. Una vez realizados los cambios deseados en la tarea podrá pulsar el botón "gestionar" y el sistema le servirá la siguiente tarea disponible en la cola de trabajo. La pantalla de la gestión de la tarea será la siguiente:

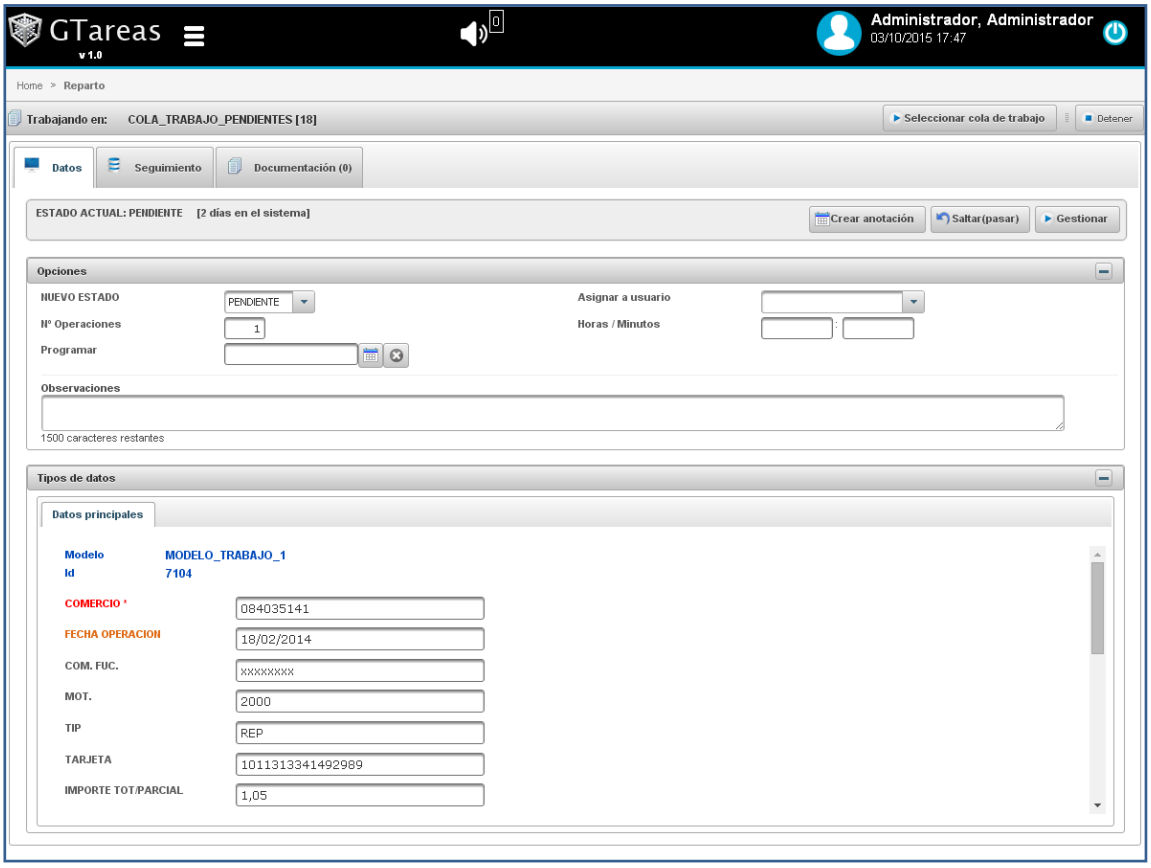

Figura 50. Aspecto visual de la pantalla "Detalle y gestión de tarea"

3.5.11 Implementación del requisito PB-0-014

En el punto 3.4.12 describimos las tareas necesarias para la implementación del requisito PB-0-014. Estas tareas son las siguientes:

- Capa de acceso a datos para la asignación de tareas a usuarios.
- Servicio de la capa de negocio que se encargará de gestionar las asignaciones de tareas a los distintos usuarios con permisos.
- Diseño de pantalla para la asignación de tareas a los usuarios.

3.5.11.1 Acceso a datos para la asignación de tareas

La clase utilizada para la asignación de tareas en la base de datos será *DAOTrabajo.java*. Esta clase y sus métodos ya ha sido descrita en el punto 3.5.5.2. En esta clase se usarán los métodos para la el registro en base datos de las asignaciones realizadas de las tareas.

3.5.11.2 Servicio de asignación de tareas

La clase utilizada para la capa de lógica de negocio en el reparto de tareas es *ServicioTrabajo.java*. Esta clase y sus métodos ya ha sido descrita en el punto 3.5.5.3. Los métodos usados de la clase *ServicioTrabajo.java* (capa de lógica de negocio de la aplicación) para el reparto de tareas, son los siguientes:

txAsignarTrabajo(Trabajo trabajo, Usuario usuario)
Método que realiza la asignación de un trabajo dado al usuario indicado.
txDesasignarTrabajo(Trabajo trabajo)
Método que realiza la desasignación de un trabajo dado que previamente estaba asignado.

Tabla 61. Métodos usados de la clase ServicioTrabajo.java

3.5.11.3 Diseño de pantalla de asignación de tareas

La capa de presentación de la funcionalidad "Asignación de tareas" está compuesta por la clase *GestionTrabajoActionBean.java* y la vista *gestionTrabajo.xhtml*.

La clase *GestionTrabajoActionBean.java* gestionará los métodos necesarios para realizar la asignación de la tarea seleccionada al usuario indicado por el usuario conectado. Un usuario podrá auto asignarse una tarea.

La vista *gestionTrabajo.xhtml*, será encargada de visualizar la opción de asignar la tarea que se está gestionando a cualquier otro usuario con permisos dentro del modelo de trabajo. Se mostrará la siguiente pantalla:

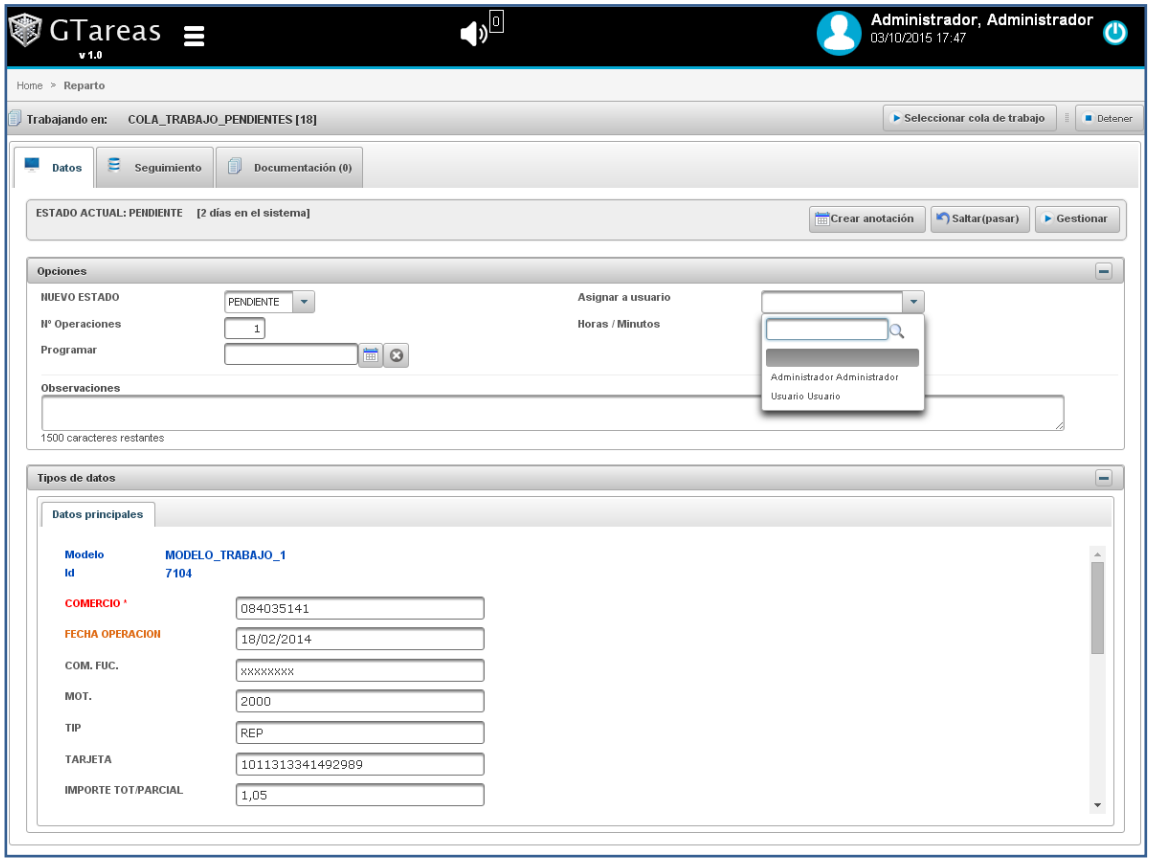

Figura 51. Aspecto visual de la pantalla "Asignación de tarea"

Capítulo 4

Plan de pruebas e implantación

4.1 Introducción

4.1.1 Plan de pruebas

El objetivo de las pruebas funcionales es validar si el comportamiento observado del software cumple o no con sus especificaciones. La prueba exhaustiva del producto requiere ejercitar todos los caminos posibles del mismo[8].

El objetivo es maximizar la producción de las pruebas, esto es, maximizar el número de errores encontrados por un número finito de los casos de prueba.

Las pruebas funcionales se consideran "Pruebas de Caja Negra", puesto que valoramos el comportamiento externo del sistema. Las "Pruebas de Seguridad" o las "Pruebas de Interoperabilidad" entre sistemas o componentes son casos especializados de las pruebas funcionales.

Las pruebas de caja negra se llevan a cabo sobre la interfaz del software, obviando el comportamiento interno y la estructura del programa. Los casos de prueba de la caja negra pretenden demostrar que:

- Las funciones del software son operativas
- La entrada se acepta de forma correcta
- Se produce una salida correcta

- La integridad de la información externa se mantiene

A continuación se derivan conjuntos de condiciones de entrada que utilicen todos los requisitos funcionales de un programa. Las pruebas de caja negra pretenden encontrar estos tipos de errores:

- Funciones incorrectas o ausentes
- Errores en la interfaz
- Errores en estructuras de datos o en accesos a bases de datos externas
- Errores de rendimiento
- Errores de inicialización y de terminación

Con el objetivo de llevar a cabo una serie de pruebas unitarias sobre la aplicación se diseñará una tabla por cada caso de prueba que se defina. En esta tabla se identificarán los siguientes campos:

- Identificador del caso de prueba: numero identificador del caso de prueba.
- Nombre del caso de prueba: nombre identificador del caso de prueba
- Descripción: breve descripción del caso de prueba aplicado.
- Fecha de Ejecución: fecha en la que se ha procedido a comprobar el caso de prueba.
- Responsable: responsable asignado a la ejecución de la prueba.
- Resultado: resultado de la aplicación de la prueba. Los valores posibles que puede tomar este campo son Acierto/Fallo.

Esta tabla, adicionalmente, contemplará una segunda sección que se utilizará con la finalidad de describir los posibles defectos encontrados. En esta sección se utilizan los siguientes campos:

- Defecto X: identificador del problema detectado dentro del caso de prueba tratado. La X representa un número secuencial que se irá incrementando a medida que se vayan encontrando problemas.
- Resumen: asociado a un determinado problema, breve descripción del mismo.
- Prioridad: campo que permite identificar la criticidad del problema detectado. Los valores posibles que puede tomar este campo son: Alta/Media/Baja.

4.1.2 Implantación

El objetivo del plan de implantación consiste en la descripción detallada de los pasos a seguir en el montaje de los servidores de aplicación y bases de datos en donde se instalará la aplicación desarrollada.

Se describirá la realización del empaquetado y despliegue de los ficheros que compondrán la aplicación, así como la ejecución de los diferentes scripts necesarios para la creación de tablas en la base de datos y la inserción de datos iniciales. Además se describirá la configuración de los diferentes archivos necesarios para la puesta en marcha de dicha aplicación.

4.2 Plan de pruebas

4.2.1 Casos de prueba del requisito PB-0-001

El requisito PB-0-001 dice que la aplicación debe ser accesible para los usuarios utilizando únicamente un navegador y dispondrá de una interfaz web que permitirá su utilización completa:

Datos de la prueba	
Identificador	001
Nombre	Acceso a la aplicación mediante interfaz web
Descripción	Se accederá a la aplicación mediante un navegador web. Se probará a introducir la url de acceso en distintos navegadores para comprobar que la aplicación es accesible.
Fecha de ejecución	26/02/2016
Responsable	Equipo de desarrollo
Resultado	ok
Defectos detectados	
Defecto1	Resumen:
	Prioridad:
	Acciones sugeridas:

Tabla 62. Caso de prueba 001. Acceso a la aplicación mediante navegador web.

4.2.2 Casos de prueba del requisito PB-0-002

El requisito PB-0-002 dice que la aplicación debe ser multiusuario, por lo que debe soportar la gestión completa de los mismos (altas, bajas y modificaciones):

Datos de la prueba	
Identificador	002
Nombre	Acceso al “Alta de Usuario”
Descripción	Solo deberán acceder a esta pantalla los usuarios con permisos. Al seleccionar la opción “Alta de usuarios” se debe mostrar un

	formulario en el que se soliciten la totalidad de datos asociados a la incorporación de un nuevo usuario. La visualización del formulario "Alta de usuarios" debe incluir los campos: username, password, e-mail, etc.
Fecha de ejecución	26/02/2016
Responsable	Equipo de desarrollo
Resultado	ok
Defectos detectados	
Defecto1	Resumen:
	Prioridad:
	Acciones sugeridas:

Tabla 63. Caso de prueba 002. Acceso al "Alta de usuario".

Datos de la prueba	
Identificador	003
Nombre	Inserción de usuario
Descripción	Se suministrarán datos correctos sobre la totalidad de los campos pertenecientes al formulario "Alta de usuarios". Tras la pulsación del botón "Guardar" se visualizará un mensaje por pantalla indicando que la inserción ha sido correcta. En listado de usuarios deberá aparecer el nuevo usuario insertado.
Fecha de ejecución	26/02/2016
Responsable	Equipo de desarrollo
Resultado	ok
Defectos detectados	
Defecto1	Resumen:
	Prioridad:
	Acciones sugeridas:

Tabla 64. Caso de prueba 003. Inserción de usuario.

Datos de la prueba	
Identificador	004
Nombre	Modificación de usuario
Descripción	Desde el listado de usuarios seleccionar el usuario a modificar. Deberá aparecer el formulario de los datos del usuario. Modificar algún dato pulsar en "Guardar". En el listado de usuarios se deberán ver los datos modificados
Fecha de ejecución	26/02/2016
Responsable	Equipo de desarrollo
Resultado	ok
Defectos detectados	
Defecto1	Resumen:
	Prioridad:
	Acciones sugeridas:

Tabla 65. Caso de prueba 004. Modificación de usuario.

Datos de la prueba

Identificador	005
Nombre	Deshabilitar usuario
Descripción	Desde el listado de usuarios de pulsará el icono de deshabilitar usuario. El estado del usuario pasará a "deshabilitado" y desaparecerá del listado.
Fecha de ejecución	26/02/2016
Responsable	Equipo de desarrollo
Resultado	ok
Defectos detectados	
Defecto1	Resumen:
	Prioridad:
	Acciones sugeridas:

Tabla 66. Caso de prueba 005. Deshabilitar usuario.

Datos de la prueba	
Identificador	006
Nombre	Acceso de usuario al sistema. Login
Descripción	Desde la pantalla de login el usuario deberá acceder introduciendo su usuario y contraseña. Si las credenciales de acceso son correctas, el usuario accederá al sistema, en caso contrario se visualizará un mensaje por pantalla indicando fallo en la autenticación.
Fecha de ejecución	26/02/2016
Responsable	Equipo de desarrollo
Resultado	ok
Defectos detectados	
Defecto1	Resumen:
	Prioridad:
	Acciones sugeridas:

Tabla 67. Caso de prueba 005. Acceso de usuario al sistema.

4.2.3 Casos de prueba del requisito PB-0-003

El requisito PB-0-003 dice que la aplicación dispondrá de varios perfiles. Se dispondrá de un módulo de administración de perfiles. Un perfil necesario es el de "Superadministrador" que configurará los parámetros de la aplicación y administrará los perfiles necesarios (en principio "Administrador" y "Usuario"). Cada uno de los perfiles creados llevará asignados los permisos necesarios:

Datos de la prueba	
Identificador	007
Nombre	Acceso al "Alta de Perfil"
Descripción	Solo deberán acceder a esta pantalla los usuarios con permisos. Al seleccionar la opción "Alta de perfil" se debe mostrar un formulario en el que se solicite el nombre que identificará al perfil. Además se deberá mostrar un listado de permisos/roles para poder seleccionar.
Fecha de ejecución	29/02/2016
Responsable	Equipo de desarrollo
Resultado	ok

Defectos detectados	
Defecto1	Resumen:
	Prioridad:
	Acciones sugeridas:

Tabla 68. Caso de prueba 007. Acceso al "Alta de perfil".

Datos de la prueba	
Identificador	008
Nombre	Asignación de perfil a usuario
Descripción	Desde el listado de usuarios seleccionar el usuario y acceder a su edición. Deberá aparecer en el formulario un listado con los perfiles del usuario y un desplegable con los perfiles a asignar. Al seleccionar un perfil del desplegable, este será asignado al usuario. El usuario al acceder al sistema deberá tener los permisos asociados al perfil asignado.
Fecha de ejecución	29/02/2016
Responsable	Equipo de desarrollo
Resultado	ok
Defectos detectados	
Defecto1	Resumen:
	Prioridad:
	Acciones sugeridas:

Tabla 69. Caso de prueba 008. Asignación de perfil a usuario.

Datos de la prueba	
Identificador	009
Nombre	Eliminación de perfil a usuario
Descripción	Desde el listado de usuarios seleccionar el usuario y acceder a su edición. Deberá aparecer en el formulario un listado con los perfiles del usuario. Al seleccionar el icono de "eliminar" del perfil, este será eliminado del usuario. El usuario al acceder al sistema no deberá conservar los permisos asociados al perfil eliminado.
Fecha de ejecución	29/02/2016
Responsable	Equipo de desarrollo
Resultado	ok
Defectos detectados	
Defecto1	Resumen:
	Prioridad:
	Acciones sugeridas:

Tabla 70. Caso de prueba 009. Eliminación de perfil a usuario.

4.2.4 Casos de prueba de los requisitos PB-0-004 al PB-0-007

Los requisitos PB-0-004 al PB-0-007 están relacionados con el módulo de gestión de modelos de trabajo, incluyendo estados, campos y permisos:

Datos de la prueba	
Identificador	010

Nombre	Acceso al "Alta de modelo de trabajo"
Descripción	Solo deberán acceder a esta pantalla los usuarios con permisos. Al seleccionar la opción "Nuevo modelo de trabajo" se debe mostrar un formulario en el que se solicite el nombre que identificará al modelo y su descripción. Además se deberá seleccionar un fichero excel donde estén los campos iniciales de los que se compone el modelo.
Fecha de ejecución	01/03/2016
Responsable	Equipo de desarrollo
Resultado	ok
Defectos detectados	
Defecto1	Resumen:
	Prioridad:
	Acciones sugeridas:

Tabla 71. Caso de prueba 010. Acceso al "Alta de modelo de trabajo".

Datos de la prueba	
Identificador	011
Nombre	Acceso al "Edición/configuración de modelo de trabajo"
Descripción	En la pantalla de administración de modelos de trabajo, al seleccionar el modelo del desplegable de modelos de trabajo se deberá ver la pantalla de administración del modelo con toda la información relacionada con estados, campos y permisos.
Fecha de ejecución	01/03/2016
Responsable	Equipo de desarrollo
Resultado	ok
Defectos detectados	
Defecto1	Resumen:
	Prioridad:
	Acciones sugeridas:

Tabla 72. Caso de prueba 011. Acceso al "Configuración de modelo de trabajo".

Datos de la prueba	
Identificador	012
Nombre	Acceso a "Configuración de estados" del modelo
Descripción	Desde la pestaña "Estados" se deberá poder modificar el estado inicial del modelo y además poder cargar un Excel para la carga de estados y sus transiciones. Al cargar el Excel, los estados cargados se visualizarán junto con las transiciones entre ellos.
Fecha de ejecución	01/03/2016
Responsable	Equipo de desarrollo
Resultado	ok
Defectos detectados	
Defecto1	Resumen:
	Prioridad:
	Acciones sugeridas:

Tabla 73. Caso de prueba 012. Acceso a "Configuración de estados del modelo".

Datos de la prueba	
Identificador	013
Nombre	Acceso a “Configuración de permisos” del modelo
Descripción	Desde la pestaña “Permisos” se deberá poder añadir o quitar usuarios tanto de la parte de administradores como de usuarios. Al añadir o quitar usuarios como administradores implicará que dichos usuarios podrán acceder o no a la administración del modelo.
Fecha de ejecución	02/03/2016
Responsable	Equipo de desarrollo
Resultado	ok
Defectos detectados	
Defecto1	Resumen:
	Prioridad:
	Acciones sugeridas:

Tabla 74. Caso de prueba 013. Acceso a “Configuración de permisos del modelo”.

Datos de la prueba	
Identificador	014
Nombre	Acceso a “Configuración de campos” del modelo
Descripción	Desde la pestaña “Configuración” se deberá poder indicar/modificar los nombres de los campos. Además para cada campo se deberá poder indicar el tipo de dato y si se desea que sea visible en la ficha, en el filtro o en el resultado de la búsqueda. Se deberá poder añadir nuevos campos.
Fecha de ejecución	02/03/2016
Responsable	Equipo de desarrollo
Resultado	ok
Defectos detectados	
Defecto1	Resumen:
	Prioridad:
	Acciones sugeridas:

Tabla 75. Caso de prueba 014. Acceso a “Configuración de campos del modelo”.

4.2.5 Casos de prueba del requisito PB-0-008

El requisito PB-0-008 dice que la aplicación dispondrá de la posibilidad de cargar masivamente las tareas desde un fichero Excel donde se indique el contenido de los campos definidos:

Datos de la prueba	
Identificador	015
Nombre	Acceso a “Carga de tareas”
Descripción	Solo deberán acceder a esta pantalla los usuarios con permisos. Al seleccionar la opción “Cargar trabajos” se debe mostrar la opción de seleccionar un Excel con el contenido de las tareas.
Fecha de ejecución	02/03/2016
Responsable	Equipo de desarrollo
Resultado	ok

Defectos detectados	
Defecto1	Resumen:
	Prioridad:
	Acciones sugeridas:

Tabla 76. Caso de prueba 015. Acceso a "Carga de tareas".

Datos de la prueba	
Identificador	016
Nombre	Carga de tareas
Descripción	Al seleccionar el Excel que contiene las tareas a cargar deberá informar al usuario que el Excel está cargando. Una vez cargado se visualizará por pantalla el mensaje de que la carga ha sido correcta. Se indicará como información el número de tareas cargadas y el número de tareas que están activas en el modelo
Fecha de ejecución	02/03/2016
Responsable	Equipo de desarrollo
Resultado	ok
Defectos detectados	
Defecto1	Resumen:
	Prioridad:
	Acciones sugeridas:

Tabla 77. Caso de prueba 016. Carga de tareas.

4.2.6 Casos de prueba del requisito PB-0-009

El requisito PB-0-009 dice que la aplicación dispondrá de la posibilidad de configurar el acceso a diversos buzones de correo. Además se podrá configurar la forma en que los correos entrantes se transforman en tareas a realizar:

Datos de la prueba	
Identificador	017
Nombre	Acceso a "Administración de buzones"
Descripción	Solo deberán acceder a esta pantalla los usuarios con permisos. Desde la pantalla "Administración de buzones" se deberá poder visualizar el listado con todos los buzones configurados en el sistema. Desde este listado deberá dar opción de editar un buzón o dar de alta uno nuevo.
Fecha de ejecución	03/03/2016
Responsable	Equipo de desarrollo
Resultado	ok
Defectos detectados	
Defecto1	Resumen:
	Prioridad:
	Acciones sugeridas:

Tabla 78. Caso de prueba 017. Acceso a "Administración de buzones".

Datos de la prueba	
Identificador	018

Nombre	Alta de buzón
Descripción	Solo deberán acceder a esta pantalla los usuarios con permisos. Desde la pantalla de alta de buzón de deberán visualizar todos los campos necesarios para la configuración de un buzón. Al dar al botón guardar aparecerá un mensaje en pantalla indicando que se ha realizado correctamente. En ese momento el nuevo buzón deberá visualizarse en el listado de buzones.
Fecha de ejecución	03/03/2016
Responsable	Equipo de desarrollo
Resultado	ok
Defectos detectados	
Defecto1	Resumen:
	Prioridad:
	Acciones sugeridas:

Tabla 79. Caso de prueba 018. Alta de buzón.

Datos de la prueba	
Identificador	019
Nombre	Reglas de creación de tareas
Descripción	Dentro de la pantalla de edición de buzones, aparecerá una pestaña de "reglas". Desde ahí se podrá crear una nueva regla donde se indicará el tipo de regla, el modelo al que pertenecerán las tareas creadas a partir de los mail recibidos.
Fecha de ejecución	03/03/2016
Responsable	Equipo de desarrollo
Resultado	ok
Defectos detectados	
Defecto1	Resumen:
	Prioridad:
	Acciones sugeridas:

Tabla 80. Caso de prueba 019. Reglas de creación de tareas.

4.2.7 Casos de prueba del requisito PB-0-010

El requisito PB-0-010 dice que el administrador de un modelo de trabajo tendrá la posibilidad de crear diversas colas de trabajo:

Datos de la prueba	
Identificador	020
Nombre	Acceso a "Administración de colas de trabajo"
Descripción	Solo deberán acceder a esta pantalla los usuarios con permisos. Desde la pantalla "Administración de colas de trabajo" se deberá poder visualizar el listado con todas las colas de trabajo de un modelo seleccionado. Desde este listado deberá dar opción de editar la cola de trabajo o dar de alta una nueva.
Fecha de ejecución	04/03/2016
Responsable	Equipo de desarrollo
Resultado	ok
Defectos detectados	

Defecto1	Resumen:
	Prioridad:
	Acciones sugeridas:

Tabla 81. Caso de prueba 020. Acceso a "Administración de colas de trabajo".

Datos de la prueba	
Identificador	021
Nombre	Creación de cola de trabajo
Descripción	Solo deberán acceder a esta pantalla los usuarios con permisos. Desde listado de colas de trabajo se deberá poder acceder a la pantalla de creación de colas. Al pulsar en "crear nueva" aparecerá un campo para indicar el nombre de la cola de trabajo. Una vez pulsado el botón "crear", se deberá poder indicar los criterios de filtrado y ordenación en función del modelo al que pertenece la cola de trabajo. Además se podrán agregar/eliminar los usuarios los usuarios que podrán trabajar en la cola y los que podrán administrar la cola.
Fecha de ejecución	04/03/2016
Responsable	Equipo de desarrollo
Resultado	ok
Defectos detectados	
Defecto1	Resumen:
	Prioridad:
	Acciones sugeridas:

Tabla 82. Caso de prueba 021. Creación de cola de trabajo.

4.2.8 Casos de prueba del requisito PB-0-011

El requisito PB-0-011 dice que el administrador de un modelo de trabajo tendrá la posibilidad de realizar un seguimiento de las tareas a través de un monitor:

Datos de la prueba	
Identificador	022
Nombre	Acceso a monitor de tareas
Descripción	Solo deberán acceder a esta pantalla los usuarios con permisos. Desde el menú lateral aparecerá la opción de acceder al monitor de tareas. Al seleccionar la opción se accederá a la pantalla del monitor donde se podrá seleccionar el modelo de trabajo a monitorizar.
Fecha de ejecución	07/03/2016
Responsable	Equipo de desarrollo
Resultado	ok
Defectos detectados	
Defecto1	Resumen:
	Prioridad:
	Acciones sugeridas:

Tabla 83. Caso de prueba 022. Acceso a monitor de tareas.

Datos de la prueba	
Identificador	023

Nombre	Visualizar monitor de tareas
Descripción	Solo deberán acceder a esta pantalla los usuarios con permisos. Una vez dentro de la pantalla del monitor de tareas de un modelo, se podrá seleccionar el rango de fechas del que se desea la información, además se podrá exportar la información a Excel.
Fecha de ejecución	07/03/2016
Responsable	Equipo de desarrollo
Resultado	ok
Defectos detectados	
Defecto1	Resumen:
	Prioridad:
	Acciones sugeridas:

Tabla 84. Caso de prueba 023. Visualizar monitor de tareas.

4.2.9 Casos de prueba del requisito PB-0-012

El requisito PB-0-012 dice que la aplicación dispondrá de un buscador de tareas dentro del sistema:

Datos de la prueba	
Identificador	024
Nombre	Acceso a buscador de tareas
Descripción	Solo deberán acceder a esta pantalla los usuarios con permisos. Al acceder a la pantalla del buscador de tareas se deberá poder seleccionar en primer lugar sobre qué modelo se desea buscar. A continuación se podrá indicar los criterios de búsqueda y ordenación sobre los campos configurados del modelo.
Fecha de ejecución	08/03/2016
Responsable	Equipo de desarrollo
Resultado	ok
Defectos detectados	
Defecto1	Resumen:
	Prioridad:
	Acciones sugeridas:

Tabla 85. Caso de prueba 024. Acceso a buscador de tareas.

Datos de la prueba	
Identificador	025
Nombre	Buscar tareas
Descripción	En la pantalla del buscador de tareas después de configurar la búsqueda se deberá poder pulsar el botón “buscar”. Una vez pulsado deberá aparecer el resultado de la búsqueda.
Fecha de ejecución	08/03/2016
Responsable	Equipo de desarrollo
Resultado	ok
Defectos detectados	
Defecto1	Resumen:
	Prioridad:
	Acciones sugeridas:

Tabla 86. Caso de prueba 025. Buscar tareas.

Datos de la prueba	
Identificador	026
Nombre	Gestión de tareas
Descripción	Desde la tabla del resultado del buscador de tareas se deberá poder seleccionar una tarea y acceder a su edición/gestión. Dentro de la edición de la tarea se deberá poder indicar los valores a los campos configurados y pulsar el botón "gestionar". Una vez gestionada la tarea se deberá regresar al listado de tareas visualizando los cambios realizados en la tarea
Fecha de ejecución	09/03/2016
Responsable	Equipo de desarrollo
Resultado	ok
Defectos detectados	
Defecto1	Resumen:
	Prioridad:
	Acciones sugeridas:

Tabla 87. Caso de prueba 026. Gestión de tareas.

4.2.10 Casos de prueba del requisito PB-0-013

El requisito PB-0-013 dice que la aplicación dispondrá de un módulo de reparto de tareas. Cuando un usuario con permisos en alguna cola de trabajo acceda a este módulo el sistema automáticamente le servirá tareas para realizar:

Datos de la prueba	
Identificador	027
Nombre	Acceso a reparto de tareas
Descripción	Solo deberán acceder a esta pantalla los usuarios con permisos. Se accederá a la pantalla de reparto de tareas y se visualizará la lista de colas de trabajo a las que se tiene acceso. Si el usuario solo tiene acceso a una cola de trabajo se le deberá mostrar la primera tarea disponible dentro de esa cola de trabajo. Si tiene varias colas de trabajo se deberá mostrar una lista de esas colas para poder seleccionar una de ellas.
Fecha de ejecución	10/03/2016
Responsable	Equipo de desarrollo
Resultado	ok
Defectos detectados	
Defecto1	Resumen:
	Prioridad:
	Acciones sugeridas:

Tabla 88. Caso de prueba 027. Acceso a reparto de tareas.

Datos de la prueba	
Identificador	028
Nombre	Gestionar tarea y recibir siguiente
Descripción	Solo deberán acceder a esta pantalla los usuarios con permisos.

	Una vez seleccionada la cola de trabajo, el sistema deberá servir una tarea al usuario y mostrarla por pantalla. Al pulsar el botón "gestionar" se registrarán los datos indicados y se deberá mostrar la siguiente tarea disponible dentro de esa cola de trabajo.
Fecha de ejecución	10/03/2016
Responsable	Equipo de desarrollo
Resultado	ok
Defectos detectados	
Defecto1	Resumen:
	Prioridad:
	Acciones sugeridas:

Tabla 89. Caso de prueba 028. Gestionar tarea y recibir siguiente.

4.2.11 Casos de prueba del requisito PB-0-014

El requisito PB-0-014 dice que la aplicación dispondrá de la posibilidad de asignación de tareas a un usuario en concreto:

Datos de la prueba	
Identificador	028
Nombre	Asignación de tareas
Descripción	Solo deberán acceder a esta pantalla los usuarios con permisos. Dentro de la edición de la tarea se deberá mostrar un desplegable con la lista de los usuarios a los que se le podrá asignar la tarea. Una vez seleccionado el usuario y pulsado el botón "gestionar" la tarea se deberá asignar al usuario elegido y solo él podrá editar dicha tarea. En el listado de tareas esa tarea deberá estar marcada como asignada.
Fecha de ejecución	11/03/2016
Responsable	Equipo de desarrollo
Resultado	ok
Defectos detectados	
Defecto1	Resumen:
	Prioridad:
	Acciones sugeridas:

Tabla 90. Caso de prueba 028. Asignación de tareas.

4.3 Implantación

4.3.1 Instalación de servidores

Para el funcionamiento de la aplicación web, se definió la necesidad de instalación de un servidor de aplicaciones y un servidor de base de datos. Para un rendimiento óptimo de la aplicación y para no disparar los costes de implantación, se decide instalar dichos servidores en una misma máquina física.

Para el servidor de aplicaciones se tomó la decisión de usar el servidor de software libre Apache Tomcat 7.0.39. Se realizará la instalación como servicio de Windows, de esta manera el servidor se iniciará en el momento que se inicie el sistema operativo. Una vez instalado el servidor de aplicaciones Apache Tomcat, se procederá a la configuración del mismo. Se deberá configurar el puerto de comunicaciones con el servidor, la reserva de memoria para uso del servidor. En la ilustración siguiente se observa la configuración del servidor de aplicaciones:

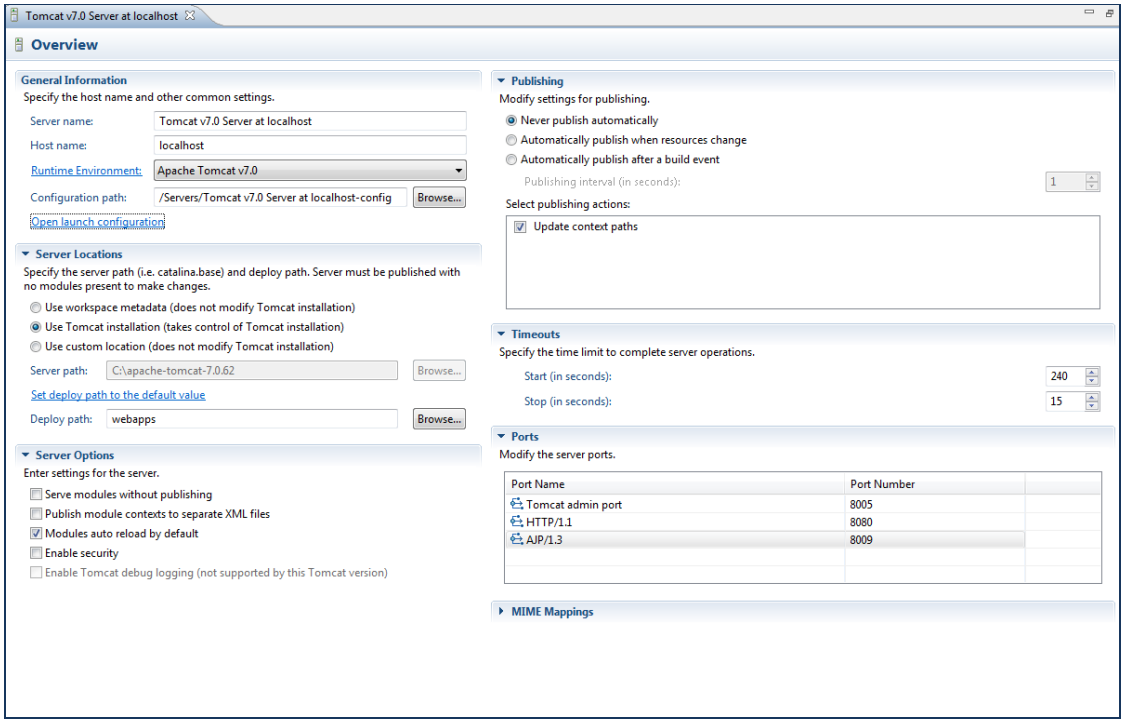

Figura 52. Configuración del servidor de aplicaciones Apache Tomcat

Como servidor de Base de Datos se decidió usar MySQL 5.6. Una vez instalado el servidor, se creará la base de datos "GTareas", vacía. Tras crear la base de datos, se creará un usuario con permisos de lectura/escritura en la base de datos para configurar la conexión en los ficheros de configuración de la aplicación.

4.3.2 Despliegue de aplicación

La aplicación se desplegará mediante un archivo WAR que empaquetará todos los ficheros que contienen el código compilado de la aplicación, las vistas y los ficheros de configuración.

Para generar el fichero WAR se utiliza la herramienta Apache Ant. Configurando un fichero XML se definen las instrucciones que debe realizar la herramienta para generar el fichero WAR. En este fichero XML se indicará la ruta donde se recogerán los ficheros de código compilado necesarios, las vistas necesarias y los ficheros de configuración, que irán todos ellos empaquetados en el mismo fichero que se llamará *GTareas.war*.

```
203
204     <!-- Create binary distribution -->
205     <target name="war" depends="build">
206         <mkdir dir="${build.dir}" />
207         <war basedir="${webroot.dir}" warfile="${build.dir}/${project.distname}.war" webxml="${webinf.dir}/web.xml">
208             <exclude name="WEB-INF/${build.dir}/**" />
209             <exclude name="WEB-INF/src/**" />
210             <exclude name="WEB-INF/web.xml" />
211             <exclude name="WEB-INF/configuracion/${project.distname}/properties*/**" />
212         </war>
213     </target>
214
```

Figura 53. Fragmento del fichero XML para generación de GTareas.war

La creación de las tablas de la base de datos se delegará en el framework Hibernate. Se configurará este framework de manera que al arrancar la aplicación, automáticamente genere todas las tablas definidas dentro de la aplicación. Para ello también definiremos un fichero properties indicando esta configuración.

```
1 # Parámetros de configuración de Hibernate
2 hibernate.dialect=org.hibernate.dialect.MySQLDialect
3 hibernate.connection.pool_size=10
4 hibernate.connection.autocommit=false
5 hibernate.show_sql=true
6 hibernate.hbm2ddl.auto=update
7
8
```

Figura 54. Fragmento del fichero de Hibernate para creación de tablas en BD

Una vez configurado todos los ficheros necesarios para el despliegue, solo restará ubicar el fichero WAR en la carpeta de despliegue del servidor de aplicaciones.

Antes de arrancar el servidor de aplicaciones, ejecutaremos los scripts necesarios en base de datos. Estos scripts insertarán en la base de datos los valores necesarios para el correcto funcionamiento de la aplicación. Se creará un usuario con todos los permisos que ejercerá de Superadministrador del sistema (root). Además se darán valores a parámetros de configuración.

Después de todo esto, se arrancará el servidor de aplicaciones, siendo ya accesible la aplicación desde cualquier navegador web.

Capítulo 5

Presupuesto

5.1 Introducción

En este capítulo se detallará el presupuesto elaborado para afrontar la implementación de la aplicación GTareas. A través de esta herramienta, se podrá llevar a cabo una completa gestión de tareas dentro de una definición de modelo de trabajo, ofreciendo la funcionalidad de reparto y asignación de las mismas.

Los puntos que se abordarán en este capítulo, referentes al presupuesto de la elaboración de la aplicación web, se exponen a continuación.

- División en fases y subfases del proyecto.
- Diagrama de Gantt.
- Resumen de costes.

5.2 División en fases y subfases

A continuación se muestra la división que se ha realizado en fases y subfases de todas las tareas a realizar en la realización de la aplicación.

Se ha definido un identificador (id) de Subfase para luego identificarla unívocamente en el diagrama de Gantt. Para cada Subfase además del identificador se indicará la duración de la misma en días.

5.2.1 División de fase "Fase Inicio"

Fase	Subfase	Id Subfase	Duración (días)
Fase Inicio	Análisis de requisitos	2	15d
	Selección de la arquitectura (framework, servidor de aplicaciones, herramientas de control y software subyacente).	3	5d
	Configuración y parametrización del servidor de aplicaciones y de la base de datos	4	2d
	Diseño de la capa de presentación de la aplicación.	5	1d

Tabla 91. División en subfases de la fase "Fase de inicio".

5.2.2 División de fase "Implementación requisito PB-0-001"

Fase	Subfase	Id Subfase	Duración (días)
Implementación requisito PB-0-001	Diseño de esquema de base de datos con las tablas necesarias para el registro de usuarios	7	2d
	Capa de acceso a base de datos tanto para el alta, la modificación y la baja de los usuarios así como la asignación de perfiles	8	1d
	Servicio de la capa de negocio que se encargará tanto de la administración como de la autenticación de usuarios	9	2d
	Diseño de pantalla de administración de usuarios.	10	1d
	Diseño de pantalla de login de acceso de usuarios.	11	1d

Tabla 92. División en subfases de la fase "Implementación requisito PB-0-001".

5.2.3 División de fase "Implementación requisito PB-0-002"

Fase	Subfase	Id Subfase	Duración (días)
Implementación	Diseño de esquema de base de datos con las tablas necesarias para el registro de perfiles.	13	2d

requisito PB-0-002	Capa de acceso a base de datos tanto para el alta, la modificación y la baja de los perfiles.	14	1d
	Servicio de la capa de negocio que se encargará de la administración de los perfiles que estarán disponibles para su asignación a los usuarios	15	2d
	Diseño de pantalla de administración de perfiles	16	1d
	Diseño de pantalla de asignación de perfiles a los usuarios	17	1d

Tabla 93. División en subfases de la fase "Implementación requisito PB-0-002".

5.2.4 División de fase "Implementación requisito PB-0-003 al PB-0-007"

Fase	Subfase	Id Subfase	Duración (días)
Implementación requisito PB-0-003 al PB-0-007	Diseño de esquema de base de datos con las tablas necesarias para el registro de modelos de trabajo.	19	2d
	Capa de acceso a base de datos tanto para el alta, la modificación y la baja de los modelos de trabajo, así como para el registro de los estados, campos y permisos.	20	2d
	Servicio de la capa de negocio que se encargará de la administración de los modelos de trabajo, sus estados, sus campos y sus permisos.	21	5d
	Diseño de pantalla de administración de modelos de trabajo.	22	2d
	Diseño de pantalla para el mantenimiento de estados por los que podrán pasar las tareas que pertenezcan al modelo de trabajo.	23	1d
	Diseño de pantalla para el mantenimiento de campos donde se guardará la información necesaria de las tareas que pertenezcan al modelo de trabajo.	24	1d
	Diseño de pantalla para el mantenimiento de los usuarios con permisos para gestionar las tareas que pertenezcan al modelo de trabajo.	25	1d

Tabla 94. División en subfases de la fase "Implementación requisito PB-0-003 al PB-0-007".

5.2.5 División de fase "Implementación requisito PB-0-008"

Fase	Subfase	Id Subfase	Duración (días)
Implementación requisito PB-0-008	Diseño de esquema de base de datos con las tablas necesarias para el registro de tareas de un modelo de trabajo.	27	2d
	Capa de acceso a base de datos para el registro de las tareas.	28	1d
	Servicio de la capa de negocio que se encargará de la lectura del fichero Excel y la adaptación de los datos recibidos.	29	2d
	Diseño de pantalla para la carga de tareas desde un fichero Excel.	30	1d

Tabla 95. División en subfases de la fase "Implementación requisito PB-0-008".

5.2.6 División de fase "Implementación requisito PB-0-009"

Fase	Subfase	Id Subfase	Duración (días)
Implementación requisito PB-0-009	Diseño de esquema de base de datos con las tablas necesarias para el registro de la configuración de acceso a buzones de correo.	32	2d
	Capa de acceso a base de datos para el registro de buzones, tareas y permisos.	33	2d
	Servicio de la capa de negocio que se encargará de la administración de los buzones, registro de tareas y administración de permisos.	34	2d
	Diseño de pantalla para la administración de buzones de correo. Además de la posibilidad de crear reglas de registro de tareas a partir de los correos y la posibilidad de administrar los permisos de los buzones.	35	1d

Tabla 96. División en subfases de la fase "Implementación requisito PB-0-009".

5.2.7 División de fase "Implementación requisito PB-0-010"

Fase	Subfase	Id Subfase	Duración (días)
Implementación requisito PB-0-010	Diseño de esquema de base de datos con las tablas necesarias para el registro de colas de trabajo y almacenar criterios de filtrado y ordenación.	37	2d
	Capa de acceso a base de datos para el registro de buzones, tareas y permisos.	38	1d
	Servicio de la capa de negocio que se encargará de la administración de los buzones, registro de tareas y administración de permisos.	39	2d
	Diseño de pantalla para la administración de colas de trabajo	40	1d

Tabla 97. División en subfases de la fase "Implementación requisito PB-0-010".

5.2.8 División de fase "Implementación requisito PB-0-011"

Fase	Subfase	Id Subfase	Duración (días)
Implementación requisito PB-0-011	Capa de acceso a base de datos para la consulta de tareas dentro de las colas de trabajo.	42	1d
	Servicio de la capa de negocio que se encargará de recoger la información necesaria para el seguimiento de tareas en las colas de trabajo.	43	2d
	Diseño de pantalla para la monitorización de las colas de trabajo.	44	1d

Tabla 98. División en subfases de la fase "Implementación requisito PB-0-011".

5.2.9 División de fase "Implementación requisito PB-0-012"

Fase	Subfase	Id Subfase	Duración (días)
Implementación requisito PB-0-012	Capa de acceso a base de datos para la consulta de tareas y acceso a sus datos.	46	2d
	Servicio de la capa de negocio que se encargará de recoger los resultados de las búsquedas con los criterios definidos.	47	4d
	Diseño de pantalla para la búsqueda de tareas mediante criterios de filtrado y mostrar resultados siguiendo criterios de ordenación. Posibilidad de acceso a los datos de las tareas resultantes.	48	2d

Tabla 99. División en subfases de la fase "Implementación requisito PB-0-012".

5.2.10 División de fase "Implementación requisito PB-0-013"

Fase	Subfase	Id Subfase	Duración (días)
Implementación requisito PB-0-013	Capa de acceso a base de datos para la consulta de tareas.	50	2d
	Servicio de la capa de negocio que se encargará de servir tareas de la cola de trabajo a la que se conecta el usuario siguiendo los criterios de filtrado y ordenación.	51	4d
	Diseño de pantalla para la conexión de los usuarios a las colas de trabajo disponibles a las que tenga permisos.	52	2d

Tabla 100. División en subfases de la fase "Implementación requisito PB-0-013".

5.2.11 División de fase "Implementación requisito PB-0-014"

Fase	Subfase	Id Subfase	Duración (días)
Implementación requisito PB-0-014	Capa de acceso a datos para la asignación de tareas a usuarios	54	1d
	Servicio de la capa de negocio que se encargará de gestionar las asignaciones de tareas a los distintos usuarios con permisos	55	2d
	Diseño de pantalla para la asignación de tareas a los usuarios	56	2d

Tabla 101. División en subfases de la fase "Implementación requisito PB-0-014".

5.2.12 División de fase "Pruebas"

Fase	Subfase	Id Subfase	Duración (días)
Pruebas	Pruebas de funcionamiento de la aplicación	57	12d

Tabla 102. División en subfases de la fase "Pruebas".

5.2.13 División de fase "Implantación"

Fase	Subfase	Id Subfase	Duración (días)
Implantación	Implantación de la aplicación	58	5d

Tabla 103. División en subfases de la fase "Implantación".

5.3 Diagrama de Gantt

El diagrama de Gantt constituye una herramienta que permite representar el cronograma de tareas llevadas a cabo en la implementación del proyecto GTareas. En el diagrama de Gantt se puede apreciar que la duración de la implementación del proyecto comprende del 01/11/2015 al 21/03/2016. Por cada tarea aparecen los recursos personales.

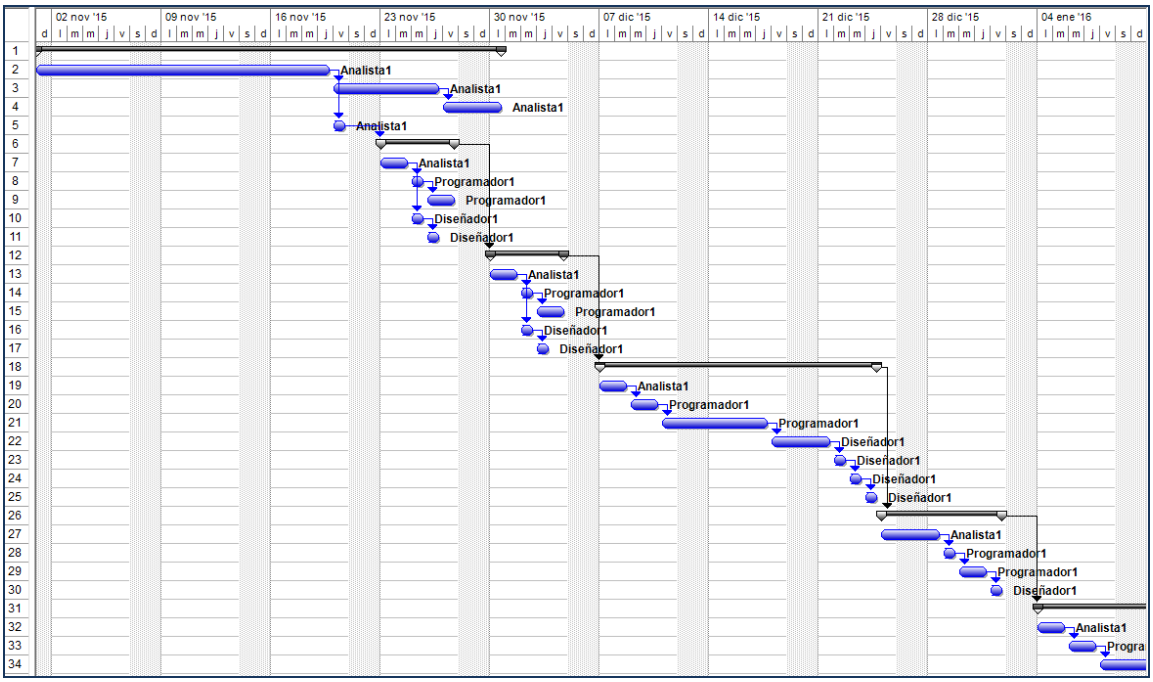

Figura 55. Diagrama de Gantt (parte 1 – subfases de 1 a 34)

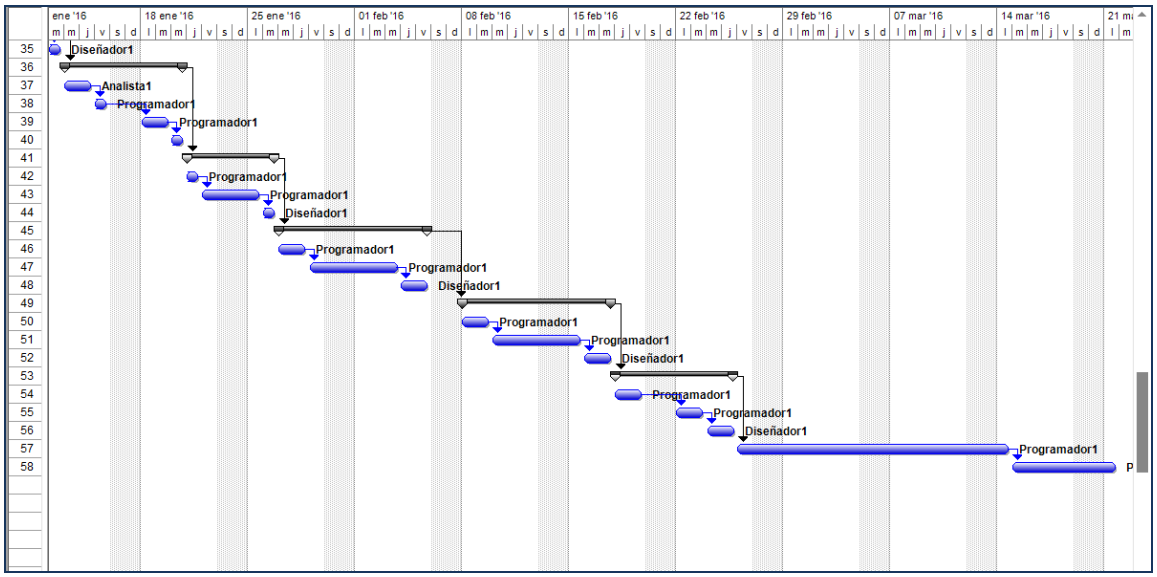

Figura 56. Diagrama de Gantt (parte 2 – subfases de 34 a 58)

5.4 Resumen de costes

En este punto se presenta el resumen de costes asociado a la implementación de la aplicación GTareas. El calendario expuesto en las figuras anteriores refleja de forma fidedigna, en media, el periodo de desarrollo de la aplicación.

La siguiente figura muestra el modelo de documento empleado en la presentación del presupuesto asociado al desarrollo de la aplicación GTareas:

Atendiendo a las cifras presentadas en la hoja de cálculo adjunta, que expone el desglose presupuestario del proyecto, se puede determinar que el presupuesto total de la aplicación GTareas asciende a la cantidad de 21.040 €. En esta cantidad no están aplicados los pertinentes impuestos indirectos. Aplicando el 21% de IVA (4.418 €), la cantidad asciende a: **25.458 €**.

Leganés a X de Octubre de 2015

El ingeniero proyectista

Fdo. Eduardo Jorge Gómez

Tabla 104. Hoja resumen del presupuesto asociado al proyecto.

A continuación se expone la hoja de cálculo utilizada para llevar a cabo la valoración económica del de desarrollo de la aplicación:

UNIVERSIDAD CARLOS III DE MADRID

Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1.- Autor: EDUARDO JORGE GOMEZ

2.- Departamento: INFORMATICA

3.- Descripción del Proyecto:

- Titulo: **APLICACIÓN WEB PARA LA GESTIÓN, REPARTO Y ASIGNACIÓN DE TAREAS GENÉRICAS**
- Duración (meses): **5,66**

Tasa de costes Indirectos: 20%

4.- Presupuesto total del Proyecto (valores en Euros):

Euros

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar - solo a titulo informativo)	Categoría	Dedicación (hombres mes) [a]	Coste hombre mes	Coste (Euro)	Firma de conformidad
					0,00	
		Analista	1,3	4.000,00	5.200,00	
		Programador	2	2.500,00	5.000,00	
		Diseñador	3,3	2.200,00	7.260,00	
					0,00	
		Hombres mes	**6,6**	**Total**	17.460,00	

[a] 1 Hombre mes = 100h.

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d]
Intel Core Duo 8Gb Memory	850,00	70	6	60	56,13
Microsoft Office Proffesional	350,00	50	6	60	17,50
Apache Tomcat v 7.0.39	0,00	100	6	60	0,00
MySQL 5.6	0,00	100	6	60	0,00
Eclipse Java EE IDE	0,00	100	6	60	0,00
Smartsheet	0,00	100	6	60	0,00
				Total	73,63

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} x C x D$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
	Total	0,00

OTROS COSTES DIRECTOS DEL PROYECTO[e]

Descripción	Empresa	Costes imputable
	Total	0,00

[e] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas,

6.- Resumen de costes

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	17.460
Amortización	74
Subcontratación de tareas	0
Costes de funcionamiento	0
Costes Indirectos	3.507
Total	21.040

Figura 57. Hoja Excel utilizada para el cálculo del presupuesto

Como puede observarse en la figura anterior, la hoja de cálculo utiliza la medida hombre/mes para establecer el coste del proyecto GTareas. Esta medida debe considerarse a efectos únicos de determinar el coste de un proyecto, pero en ningún caso debe ser considerada como una medida de alcance del proyecto.

El coste total del proyecto, al considerar una tasa de costes indirectos del 20%, asciende a la cantidad de 21.040 € (Sin IVA). Al aplicar el 21% de IVA (4.418 €), la cantidad asciende a: 25.458 €.

Capítulo 6

Conclusión

6.1 Introducción

En este capítulo se realizará una exposición de las conclusiones extraídas en el proyecto, partiendo de la base de los objetivos planteados al inicio. Adicionalmente, se comentarán las dificultades encontradas a lo largo de la realización del proyecto.

6.2 Conclusiones

Los objetivos planteados inicialmente, junto con el detalle de su cumplimiento aparecen comentados a continuación:

- Presentar una interfaz amigable que permita la utilización simultánea de la aplicación desarrollada a usuarios con diferentes grados de conocimiento sobre la materia, este objetivo se considera cumplido gracias al diseño realizado con el framework Primefaces 5.2.

- Permitir el acceso a la aplicación desde cualquier dispositivo con un navegador web disponible. Al ser una aplicación web desplegada en un servidor de aplicaciones Tomcat que es un contenedor web, el objetivo se cumple de manera satisfactoria.

- Permitir el despliegue de la aplicación en entornos distribuidos utilizando el modelo cliente-servidor. Al igual que el objetivo anterior, gracias al acceso mediante navegador web este objetivo se cumple.

- Ofrecer un alto nivel de parametrización de la aplicación para alcanzar el objetivo de que las tareas sean *genéricas*, es decir que puedan implementar cualquier situación real que se nos plantee. Este objetivo se cumple al utilizar el modelo de trabajo como configurador de tareas. Al definir los modelos de trabajo de forma tan configurable, obtenemos las tareas genéricas.

- Permitir distintos niveles de acceso implementando un sistema de permisos de usuarios basado en perfiles. Este objetivo se cumple gracias a la configuración de los perfiles. Se podrán crear tantos perfiles como sean necesarios con los permisos pertinentes.

- Permitir a los usuarios con los suficientes permisos configurar la aplicación para que se realice un control de las tareas personalizado. Permitiendo a los *administradores* configurar tantos *modelos de trabajo* como sean necesarios. Cada uno de estos *modelos de trabajo* con la información que se desee. Dentro de la filosofía de la aplicación queda cumplido este objetivo, se podrán definir tantos modelos de trabajo que sean necesarios para el control de las tareas a realizar.

- Permitir a los usuarios con los suficientes permisos configurar la aplicación para que se realice el reparto de tareas basado en criterios de búsqueda y ordenación configurados desde la aplicación. Este objetivo se cumple gracias al sistema de reparto de tareas implementado dentro del modelo de trabajo al definir tantas colas de trabajo como se necesitan según los criterios de manejo de las tareas.

- Permitir a los usuarios *administradores* de los distintos modelos de trabajo llevar un control de la actividad mediante monitores de control. Este objetivo también se ha cumplido al implementar un módulo de monitorización de tareas.

- Ofrecer la posibilidad de indicar el estado en el que queda la tarea después de su gestión para llevar un seguimiento pormenorizado de la misma. Dentro del detalle de cada tarea y su seguimiento podemos saber en todo momento en qué estado se encuentra y la trazabilidad de todos los estado por los que ha pasado.

- Ofrecer la posibilidad de asignación de tareas tanto masiva como individual a usuarios específicos. Este objetivo también queda cumplido al poder asignar tareas a cualquier usuario con permisos desde dentro de la propia ficha de la tarea.

6.3 Conclusiones personales

Las aportaciones, a nivel personal, durante la realización del proyecto fin de carrera han sido múltiples y engloban aspectos diferenciados que incluyen la ampliación de conocimientos de metodologías de desarrollo del software y evaluación de herramientas de desarrollo y tecnologías empleadas actualmente en la implementación de aplicaciones.

A parte de las anteriores aportaciones, me gustaría destacar la recompensa que he recibido al realizar este proyecto fin de carrera. Después del gran esfuerzo realizado es un placer tener la sensación del trabajo cumplido.

Glosario

AJAX	*Asynchronous JavaScript And XML*
API	*Application Programming Interface*
DAO	*Data Access Object*
GTareas	*Gestor de tareas*
IDE	*Integrated Development Environment*
JAVA EE	*Java Enterprise Edition*
JDBC	*Java Data Base Connectivity*
JDK	*Java Development Kit*
JSF	*Java Server Faces*
SGBD	*Sistema Gestor de Base de Datos*
SGBDR	*Sistema Gestor de Base de Datos Relacional*
SQL	*Structured Query Language*
XHTML	*eXtensible HyperText Markup Language*
XML	*eXtensible Markup Language*
URL	*Uniform Resource Locator*
WAR	*Web Application Archive*

Referencias

[1] LanceTalent, «LanceTalent,» [En línea]. Available: http://www.lancetalent.com/blog/las-10-mejores-herramientas-para-la-gestion-de-tareas/. [Último acceso: Septiembre 2015].

[2] «proyectosagiles.org,» [En línea]. Available: http://proyectosagiles.org/que-es-scrum/. [Último acceso: Septiembre 2015].

[3] «Codejobs,» [En línea]. Available: https://www.codejobs.biz/es/blog/2014/01/28/la-programacion-por-capas. [Último acceso: Septiembre 2015].

[4] «Wikipedia-DAO,» [En línea]. Available: https://es.wikipedia.org/wiki/Data_Access_Object. [Último acceso: Septiembre 2015].

[5] «arquitecturaencapas.blogspot.com.es,» [En línea]. Available: http://arquitecturaencapas.blogspot.com.es/2011/08/arquitectura-3-capas-programacion-por.html. [Último acceso: Septiembre 2015].

[6] «El blog del analista funcional,» [En línea]. Available: http://analistafuncional.blogspot.com.es/2012/10/metodos-para-analisis-funcional.html. [Último acceso: Septiembre 2015].

[7] «Elementos de UML,» [En línea]. Available: https://docs.kde.org/stable4/es/kdesdk/umbrello/uml-elements.html. [Último acceso: Septiembre 2015].

[8] B. P. Lamancha, «Estrategia de gestión de las pruebas funcionales en el Centro de Ensayos de Software,» *REICIS, Revista Española de Innovación, Calidad e Ingeniería del Software,* vol. 3, nº 3, 2007.

[9] «proyectosagiles.org,» [En línea]. Available: http://proyectosagiles.org/lista-requisitos-priorizada-product-backlog/. [Último acceso: Septiembre 2015].

[10] «proyectosagiles.org,» [En línea]. Available: http://proyectosagiles.org/ejecucion-iteracion-sprint/. [Último acceso: Septiembre 2015].

[11] V. F. Alarcón, Desarrollo de sistemas de información: una metodología basada en el

modelado, Barcelona: Edicions UPC, 2006.

[12] «W3C España,» [En línea]. Available: http://www.w3c.es/Divulgacion/GuiasBreves/HojasEstilo. [Último acceso: Septiembre 2015].

[13] «Wikipedia - Getting_Things_Done,» [En línea]. Available: https://es.wikipedia.org/wiki/Getting_Things_Done. [Último acceso: Septiembre 2015].

[14] «Wikipedia - Java Database Connectivity,» [En línea]. Available: https://es.wikipedia.org/wiki/Java_Database_Connectivity. [Último acceso: Septiembre 2015].